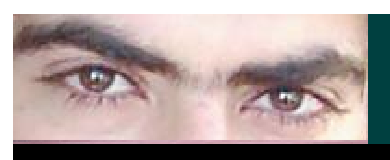


Mahmoud Fayed Book

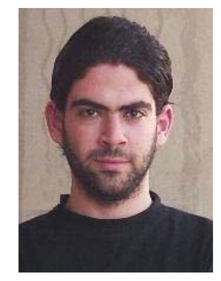
اعمق اسرار البرمجة



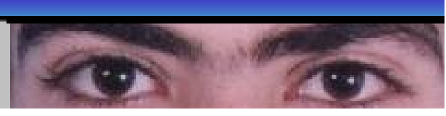
الجزء الاول - الكتاب من ستة اجزاء

هذا الكتاب :-

- يخاطب المبرمجين المحترفين
- هام للعلماء والباحثين فى علم البرمجة
- يفتح الطريق نحو مستوى جديد من الاحتراف
- المحتوى فى المجمل فريد



Programming Secrets



إهداء

*** الى والدتى الحبيبة التى طالما تحملت الالم من اجل سعادتى.**
*** الى كل عاشق للعلم ومحب للمعرفة**

شكر خاص :-

- م : إبراهيم على فايد
  - والدى ا بيب و اول من ادخل ا عا الكمبيوتر والـ ة
- م : مد إبراهيم فايد
  - اخى العزيز الذى طا ا مد ا يد العون تابعة التكنولوجيا ا تطورة
- د. : فادى سعيد إبراهيم فايد
  - ساعد كث ا ا صول عل العديد من الـ امج والتطبيقات النادرة
- م : دى مد أبو العطا
  - اظفر قابلته شخصيا حـ الان – ولكـ تعلمت من كتبه الكث
- م : مد سعود (مهدى)
  - صديق هام للاسرة – طا ا امد بالكث من ا علومات
- د. م : إيهاب عزيز خليل
  - معلمى الاول ا امعة – اول من علم ماهو البحث العلمى
- د. م : صلاح شعبان
  - تعلمت منه الكث صوص بر ة الكائنات
- م : أ د عبدالعظيم غزية
  - تعلمت منه اساسيات لغة **JAVA**
- م : أ د الكورا
  - تعلمت منه اساسيات الالكترونيات – وكيفية التعامل ا ياة ا امعية
- م : أ د دى عرفة
  - طا ا امد يد العون فسم هندسة وعلوم ا اسبات
- د. : عمرو طلبة
  - اول من ارشد ا ط بر ة العميل **AOP** ولذلك تاث ا كب ا على.
- م : مد ا دهد
  - اظفر قابلته – ولكن كتابه عن **VFP** شجع على كتابة الكتب الـانية
- **Mr. Ferns Paanakker**
  - عملنا سويا (مشروع **FGLIB**) ومنه تعلمت ادارة ا شاريع مفتوحة ا صدر
- **Yasushi Kambayashi, PhD**
  - افاد بالكث من الاراء (مشروع **DoubleS**) واستفدت من ا ائه.

"و النهاية لا انسى ان اشكر كل من د شبكة الانترنت
با قالات
ا فيدة ويشارك ا نتديات بالاراء الصائبة.
و الواقع لا تسع مئات الصفحات حـ اذكر ا ائهم – ولكنهم
بالفعل يستحقون ذلك لانم كانوا دائما اسرع طريق ل ا شاكل
الطارئة وخـ
مرشد للحصول على العديد من ا راجع ا يدة."

معلومات حول الكاتب


السيد : محمود سمير إبراهيم فايد

- طالب بكلية الهندسة الالكترونية


- (جامعة المنوفية – جمهوية مصر العربية)

msfclipper@hotmail.com


- قسم هندسة وعلوم الحاسب. مواليد ١٩٨٦\١٢\٢٩

- من الشباب الجدد الذين تربوا منذ نعومة اظفارهم على استخدام الحاسب وبرمجته – فقد بدا تعلم البرمجة عام ١٩٩٧ وهو فى العاشرة من العمر – وخلال اربعة سنوات اتقن البرمجة – ومنذ بداية عام ٢٠٠٠ اتجه الى البحث العلمى فى تخصصات متنوعة داخل علم البرمجة – وخلال رحلة بحث طويلةاستمرت ٥ سنوات تمكن من احتراف تصميم وتطوير نظم ادارة الاحداث Event-Driven Systems بالاضافة الى نظم ادارة البيئة الرسومية GUI Management Systems ومن ثم انطلق الى عالم نمط البرمجة Programming Paradigm والذى يعد قلب و نقطة انطلاق وتطور علم البرمجة وخلال عام ٢٠٠٦ تمكن من ابتكار نمط برمجة جديد متطور (نمط برمجة الخادم الممتاز Super Server Paradigm) والذى يطلق عليه دبل اس DoubleS وهذا النمط ياخذ فى الاعتبار هياكل البيانات المعقدة ونظام ادارة الاحداث وتطبيقات الزبون-الخادم وغيرها العديد من سمات التطبيقات المتطورة. وخلال رحلة العمل فى البرمجة والابحاث العلمية – اتقن العديد من اللغات وهى سى وكليبر وفيجوال فوكس برو و اكس هاربور C,CA-Clipper,VFP,xHarbour & Harbour/MiniGUI – وامتلك خبرة جيدة فى لغات اخرى وهى اسمبلى و فيجوال بيسك وجافا.

الدافع وراء هذا الكتاب

"فى البداية حتى لا استقبل اى رسائل هجومية بخصوص عنوان الكتاب (اعمق اسرار البرمجة) فانى ورغم رحلة عمل ودراسة ١٠ سنوات فى التخصص – لا اعنى بهذا الكتاب اننى عالم فى البرمجة وانما اعنى ان لدى شىء من المستحب بالنسبة لى ان اصله للاخرين واتمنى النجاح فى ذلك."لما شعر الكاتب انه قد وصل الى مرحلة جيدة – وجد ان تلك الرحلة كان من الممكن ان تكون اقصر بكثير اذا توفرت العديد من الكتب والمراجع العربية – ولا ينفى ذلك تقدير لمجهودات الاخوة من الكتاب العرب الذين قد استفدت منهم استفادة كبيرة – ولكن ان جميع الكتب التى اطلعت عليها كانت تنصب فى مجال استخدام تكنولوجيا البرمجة ولم تتطرق الى ماوراء تصنيع تكنولوجيا البرمجة – فنحن احق بان نصنع التكنولوجيا التى نستخدمها فى البرمجة او نساهم فى تطورها على الاقل – ولا نكتفى فقط بانتظار التكنولوجيا التى تاتى من العالم الغربى حتى نستخدمها.ان الكتب العربية تتجه الى التطبيقات التجارية (انظمة قواعد البيانات) بينما هذا الكتاب يسلك اتجاه اخر – انه يتجه الى النظم بمختلف انواعها مما يفتح عين القارئ على العديد من اسرار تصنيع البرمجيات.ان تصنيع تكنولوجيا البرمجيات ترتكز على الابحاث العلمية ثم التصميم ثم البرمجة – انها عملية تمر على ثلاثة مراحل – وتتطلب ثلاثة مستويات من العلم وهى (الذكاء الاصطناعى AI - وبرمجة العتاد " Low Level Programming " وبرمجة التطبيقات ذات المستوى الرفيع " High Level Programming ") يكاد ان معظم المبرمجين فى العالم العربى هم High Level Programmer او Developers مطورين والقليل جدا من يدخل الى عالم Low Level Prorgamming وينجح فيه ويصبح منتجا – والنادر جدا من يدخل عالم الذكاء الاصطناعى AI كمبرمج ويستطيع ان يبدع فيه – والسبب ليس نقص فى العقول وانما عدم وضوح للطريق وعدم توفر كتب عربية جيدة فى هذه التخصصات. هذا الكتاب هو مجرد محاولة لفتح الباب للكتابة التى تهدف الى خلق مستوى جديد من الاحتراف لدى المبرمجين العرب – وهى رسالة تشجيعية من مبرمج بسيط متواضع – الى المبرمجين المحترفين فى الوطن العربي حتى يتوقفوا قليلا عن كتابة الاسطر البرمجية – ويبدوا فى كتابة الاسطر التعليمية التى تتدخل المبرمجين العرب الى عالم برمجة النظم بمختلف انواعها.

اعمق اسرار الـ ة

هذا الكتيب الصغ للم الذين يقومون بتطوير التطبيقات التجارية ويرغبون رفع مستوى الكفاءة لديهم وذلك بان يضعوا اقدامهم على طريق بر ة النظم ختلف انواعها وال تستلزم مهارات لا توجد ا اجة اليهـا عند تطوير التطبيقات. يركز الكاتب على كيفية ادارة النظام من حيث (وذج سـ العمليات – ط الـ ة - واجهة الـ نامج – ادوات التطوير) ويقدم الكاتب حلول يفترض ان لا تتوفر اى كتاب اخر لان يع ا لول عبارة عن ربة ذاتية من ا ولف وهذا يع ان ا ولف لا نع من استقبال اى اراء علمية مفيدة. كمـا ـاول الكاتب التركيز على مفهوم عدم الاعتماد على ادوات التطوير سواء كانت (نظام تشـغيل – لغـات بر ـة) فا مج ا ترف ب ان تلك فنيات الـ ة لاكثر من نظام وباكثر من لغـة لان الادوات وسـيلة لتحقيـق الاهداف وليست عقيدة. كما يتعرض الكاتب لموعة لاباس با من ادوات تطوير التطبيقات ا تطورة حـ يفتح افق القارىء على احدث وسائل تطوير تكنولوجيا الـ يات بالاضافة ا ان الكتيب يتطرق ا التطبيقـات ا تطورة و اتا ا ختلفة من حيث (واجهة ا ستخدم – تطبيقات الزبون وا ادم – تطبيقات الانترنـت) كمـا يشمل الكتيب عرض فهوم يط التطوير ا اص (الذى يقوم ا مج ببنائه بنفسه) ا يسمح بزيادة الانتاجيـة (انتاج كم كبـ من التطبيقات فترة قصـ ة). ويشمل الكتيب كم هائل من ا علومات العلمية ا دعمة بالعديد من الامثلة العملية وبذا يتحقق طر ا عادلة الصعبة من حيث خلق جيل جديد من ا مدعم باساس علمى قوى يتيح له فرصة الابتكار و تلك خـ ة عملية تكون بالنسبة له اساس قوى واجهة ديات السوق وتغـ اتـه ا ختلفه. وا دير بالذكر ان الكتيب لا يقف عند نظام تشغيل معـ او لغة بر ة ددة بل يضرب الامثلة بلغات بر ة تلفة فهناك مثلا امثلة بلغة سى C وامثلة اخرى بلغة Visual FoxPro وهناك تطبيقـات يـتم عملها ت دوس Dos واخرى ت نظام النوافذ Windows وهكذا. ادوات التطوير ا سـتخدمة الكتيب بعضها ا freeware ومفتوح ا صدر "Open source " و كن ا صول عليها مـن خلال الانترنت و بعضها غـ ا " Shareware "وسوف تاج لشرائه من السوق او من الانترنـت من خلال مواقع الشركة ا نتجة. كما يشـمل الكتـاب جـزء يتعلـق بـ ـة العتـاد Low Level Programming حـ يكتمل هدف الكتيب ويصبح لدى القارى معرفة جيدة بالاركان الاساسية لعلم الـ ة.

و النهاية يتعرض الكتاب لكيفية اعداد الا اث العلمية الـ تتعلق بالـ ة

المحتويات

الجزء الاول :- تصميم و برمجة النظم Systems Design & Programming

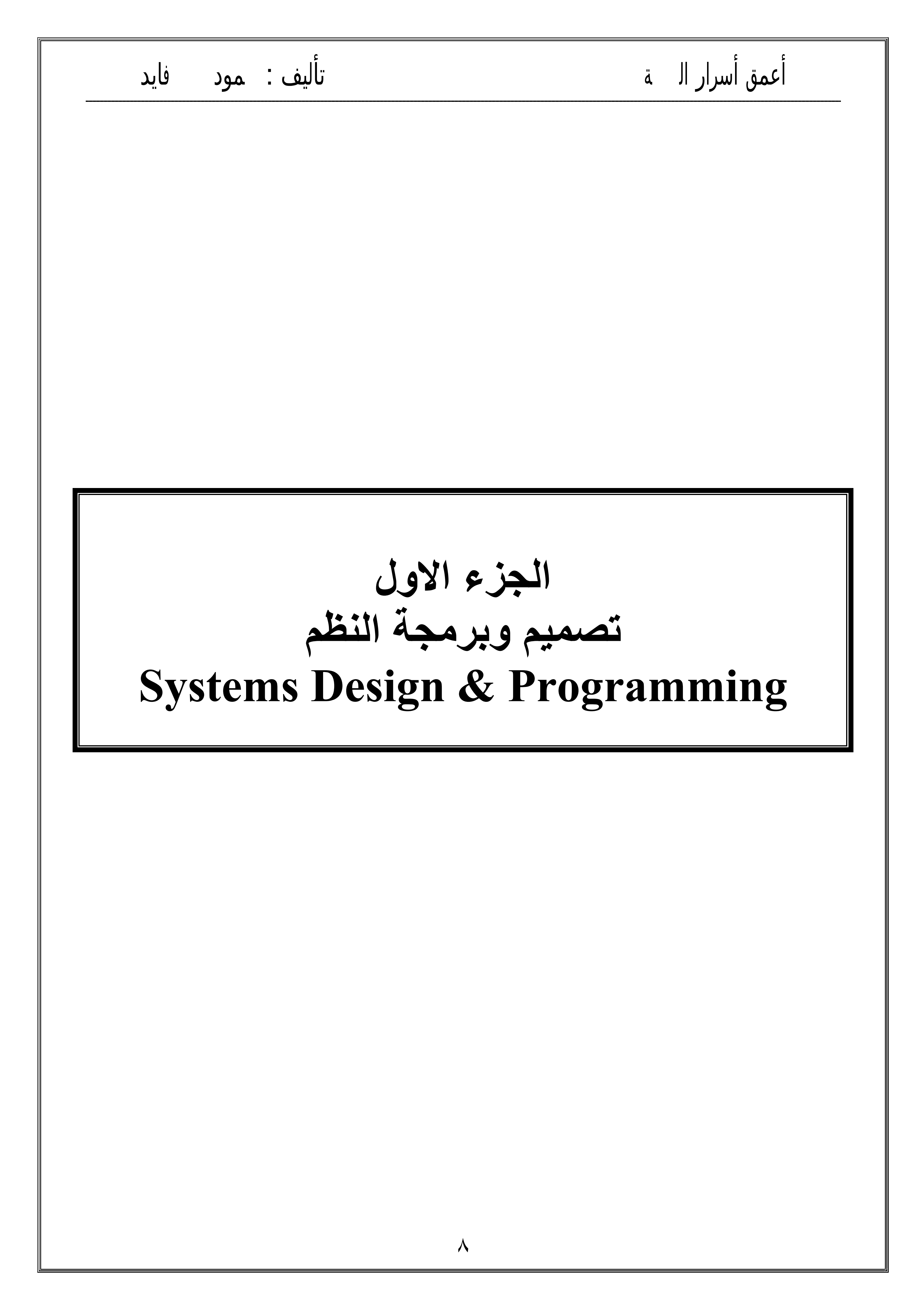
الجزء الثانى :- تكنولوجيا تطوير التطبيقات المتطورة

الجزء الثالث :- سمات التطبيقات المتطورة Modern applications Features

لجزء الرابع :- ادوات المبرمج المحترف Professional Programmer Tools

الجزء الخامس :- برمجة العتاد Low Level Programming

الجزء السادس :- البرمجة والابحاث العلمية Researches In Programming



الجزء الاول
تصميم وبرمجة النظم
Systems Design & Programming

مقدمة هامة :-

اهلا بك اخى الحبيب فى مغامرات البرمجة – بالفعل هى كذلك الان اذا ما حاولت الخروج عن تطوير التطبيقات التجارية التى اعتدت عليها ورغبت فى تطوير النظم – مبرمج النظم هو اعلى بكثير من مبرمجين التطبيقات فى المستوى العلمى حيث كلمة نظام فى حد ذاتها تعنى

نظام = كم كبير من العلم + كم كبير من التجارب + نسبة عالية من الذكاء

والنظام قد يكون نظام تشغيل كمبيوتر مثل Unix, Dos, Windows, Mac, OS/2, Linux,...etc ومبرمج النظم قد يعمل منفردا وهذا نادرا جدا جدا وقد يعمل ضمن مجموعة صغيرة او مع عدد كبير من المجموعات المصنفة الى مجموعات رئيسية وفرعية وهكذا .

كما ان النظام قد يكون جزء من نظام اخر فمثلا نظام التشغيل مثلا عبارة عن مجموعة من النظم التى تعمل معا (النواة + نظام ادارة العمليات + نظام الملفات + نظام ادارة الذاكرة + ...وهكذا)

ولا يقتصر مصطلح "نظام" على انظمة تشغيل الحاسب فقط – بل يمتد ليشمل البرمجيات الغير مالوفة – او الغير متعارف على طريقة محددة لبرمجتها فمثلا ادوات التطوير الخاصة بلغات البرمجة مثل مصمم النماذج والتقارير وغيرها يمكن اعتبارها انظمة.

النظام هو السوفت وير الذى يدير طريقة عمل شى معين داخل بيئة عمل البرامج

لتعريف مصطلح نظام
١ – هو السوفت وير الذى يحتاج الى ذكاء + تجربة وخطا حتى يتم عملية تصميمه
٢ – يتعرض لعوامل كثيرة توثر على الاستقرار والكفاءة فى العمل
٣ – غالبا ما يقدم خدمات لبرامج اخرى
٤ – ينتج تصميم النظام نتيجة بحث علمى او مجهودات كبيرة تستمر لفترة طويلة

كما ان النظام عند برمجته يستند الى المبرمج ومهارته بنسبة ٩٩% اكثر من كفاءة اللغة المستخدمة فى تطوير النظام – ولهذا نجد ان الكثير من الانظمة المتطورة تم تطويرها بلغات قديمة صعبة ومعقدة.

من مشكلات برمجة النظم انك تبدا غالبا من الصفر من اجل السيطرة على كل شئ فى بيئة العمل والعوامل الثلاثة الاساسية التى توثر فى ذلك هى

١ – نموذج سير التعليمات
٢ – نمط البرمجة

٣ – واجهة البرنامج

واذا تمكنت من التحكم فى هذه العوامل الثلاثة (من خلال برمجتها من الصفر) فانك تمتلك العناصر الاساسية لبرمجة اعقد النظم وهذا ليس باليسير ابدا ولكنه ليس مستحيل وتذكر اننا هنا نناقش الفكر فى برمجة النظم مبتعدين عن بعض التفاصيل التى تتعلق بعلم برمجة العتاد .Low Level programming-Hardware programming

اولا : نموذج سير العمليات

ان برمجة الحاسب من خلال طريقة كتابة الاكواد (يوجد طرق اخرى للبرمجة بدون اكواد مثل استخدام المصمم Designer والمعالجWizard) تتضمن وجود اساليب مختلفة للتحكم بطريقة عمل النظام او ما يعرف بنموذج سير العمليات – من المعروف ان التعليمات يتم تنفيذها واحدا تلو الاخر ولكن المقصود *بنموذج سير العمليات هو كيفية التحكم برمجيا بترتيب تنفيذ العمليات*.

ان نموذج سير العمليات لا يقصد به على الاطلاق تركيبات التحكم Control Structure وانما يقصد به كيفية توظيف تركيبات التحكم للحصول على ترتيب معين لتنفيذ العمليات – والفرق بين نموذج سير العمليات وتركيبات التحكم ان تركيبات التحكم تشمل جزء معين من التعليمات داخل النظام لكن نموذج سير العمليات يشمل النظام كاملا.

تركيبات التحكم مثل While loop & if statement وهكذا.

نموذج سير العمليات داخل نظام مايكروسوفت وندوز Microsoft Windows هو نظام الحدث Event Model حيث نتيجة لحدوث حدث معين يتم تغير سير العمليات وذلك بتنفيذ التعليمات المرتبطة بهذا الحدث

نموذج سير العمليات داخل نظام مايكروسوفت دوس Microsoft Dos هو نظام شكلى Modal Model حيث ينتظر النظام من المستخدم حدث معين وهو الضغط على مفتاح الادخال Pressing Enter key in keyboard مثلا لتنفيذ امر معين.

اى فى Modal Model ينتظر النظام حدث من المستخدم لتتابع سير العمليات.

ان تحديد نموذج سير العمليات داخل التطبيقات يختلف بكثير عنه فى داخل النظم لان نموذج سير العمليات داخل التطبيقات يفترض فى البداية انه يرتبط بكل من نظام التشغيل المستخدم + لغة البرمجة فمثلا نظام Windows يقدم Event Model بصورة مباشرة على العكس مع نظام Dos الذى تم بنائه على نظام Modal Model.

ان تحديد نموذج سير العمليات داخل النظم يعد خيار المبرمج – لانه يقوم ببناء كل شى من الصفر ولا يعتمد على نظام التشغيل او لغة البرمجة فى اتاحة مثل هذا الامر.

اى انه فى برمجة التطبيقات العادية انت لا تفكر فى نموذج سير العمليات وانما فقط تفكر فى استخدامه – لكن فى برمجة النظم انت تفكر فى نموذج سير العمليات لانك من سوف يبرمجه من البداية.

ثانيا : نمط البرمجة

ان نمط البرمجة من المفترض ان توفره لغة البرمجة عند برمجة التطبيقات العادية ومن امثلة انماط البرمجة نمط البرمجة الهيكلية Structure Programming ونمط برمجة الكائنات Object Oriented Programming (OOP) ويوجد انماط اخرى ليست مشهورة مثل نمط برمجة العميل Agent Oriented Programming ونمط برمجة اللغات الموجه Language Oriented Programming ونمط برمجة الخادم الممتاز وهو من ابتكار مولف هذا الكتيب ويمكن الاطلاع عليه من خلال الموقع http://www.sourceforge.net/projects/doublesvsoop

وعند تطوير بعض النظم الخاصة جدا قد تنشا الحاجة الى تطوير نمط برمجة خاص ولتنفيذ ذلك هناك عدة طرق
١ – انشاء لغة برمجة جديدة (اصعب طريق)
٢ – انشاء محيط تطوير (مهمة شاقة)
٣ – عمل مكتبة باستخدام Preprocessor (غير متاح بكفاءة فى كل اللغات)

ولكن الاكثر شيوعا هو استخدام نمط البرمجة المتاح داخل اللغة التى يتم تطوير النظام باستخدامها وذلك له مزايا منها عدم الحاجة الى تعلم نمط برمجة جديد من قبل الكم الهائل من المبرمجين المشتركين فى تطوير النظام.

ثالثا : واجهة البرنامج

دعنا نقول ان واجهة البرنامج تكون محددة بصورة مباشرة بامكانيات نظام التشغيل او لغة البرمجة او ادوات التطوير المتوفرة – لكن عند تطوير النظم لابد للمبرمج المحترف ان لا يعترف بمثل هذه القيود وان يمتلك السيطرة الكاملة على واجهة النظام اما باستخدام ادوات غاية فى التطور والمرونة – او بعمل واجهة النظام من البداية (من الصفر) وهذا بالتاكيد يتطلب مراعاة عوامل اخرى حيث ان واجهة النظام لابد ان تتوافق مع نموذج سير العمليات فمثلا الواحهة المصممة لتعمل فى نظام سير عمليات كلى Modal Model لا تعمل فى نظام سير عمليات مبنى على الحدث Event Model والعكس صحيح.

الخلاصة :

تطوير النظم يشمل العديد من العوامل وهذا الكتيب يناقش اعقد ثلاث عوامل من الناحية الفكرية وهى نموذج سير العمليات ونمط البرمجة بالاضافة الى واجهة النظام

– والجدير بالذكر ان هذه العوامل ترتبط معا بارتباط شديد غاية فى التعقيد فمثلا نموذج سير العمليات يحتاج واجهة مخصصة تتعامل معه كما انه لابد من استخدام نمط برمجة مناسب لتطوير كل من نموذج سير العمليات وواجهة النظام بالاضافة الى مهام النظام الاخرى.

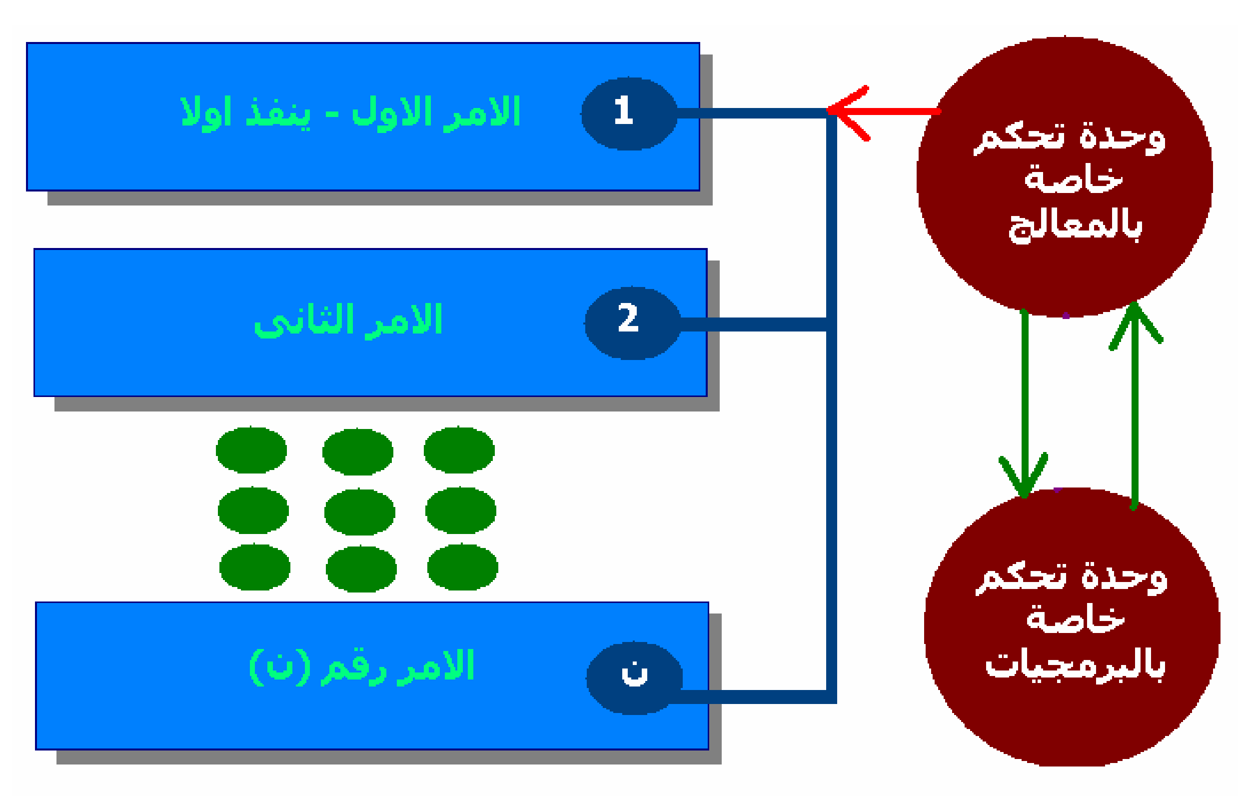
يقدم لك الكتيب

- كيف تكتب نموذج سير العمليات الخاص بالنظام
- كيف تستخدم نمط البرمجة باعلى كفاءة و كيف تبتكر نمط البرمجة الخاص بك.
- كيف تكتب واجهة النظام الخاصة بك.

الباب الاول
نموذج سير العمليات

مفهوم سير العمليات فى البرمجة :-

ان العمليات يتم تنفيذها واحدة تلـو الاخـرى - ويمكـن نقـل التنفيـذ مـن نقطـة لاخـرى JUMP سـواء نقطة علوية سـبق المرور عليها Backward او نقطة سـفلية لم نمـر عليهـا من قبل اثناء تنفيذ البرنامج Forward ويتم تحديد اين يتم القفز من خلال جمل التحكم او مايسـمى بتركيبـات الـتحكم Control Structure مثـل جمـل القـرار If Statement و Switch و Case وجمل التكرار مثل Loops بمختلف انواعها مثـل While loop و Do until و For loop وغيرها الكثير - انظر الشكل التالى رقم (١)



شكل رقم (١) : التحكم بسير العمليات من خلال المعالج والبرمجيات معا.

فى الحقيقة ان استخدام جمل التحكم لنقل التنفيذ من نقطة لاخرى يندرج فقط تحت نقطة وحدة التحكم الخاصة بالمعالج حيث ان الانتقال من نقطة لاخرى يكون حالة خاصة ومحددة او بشرط محدد ومعروف مسبقا من قبل مصمم البرنامج

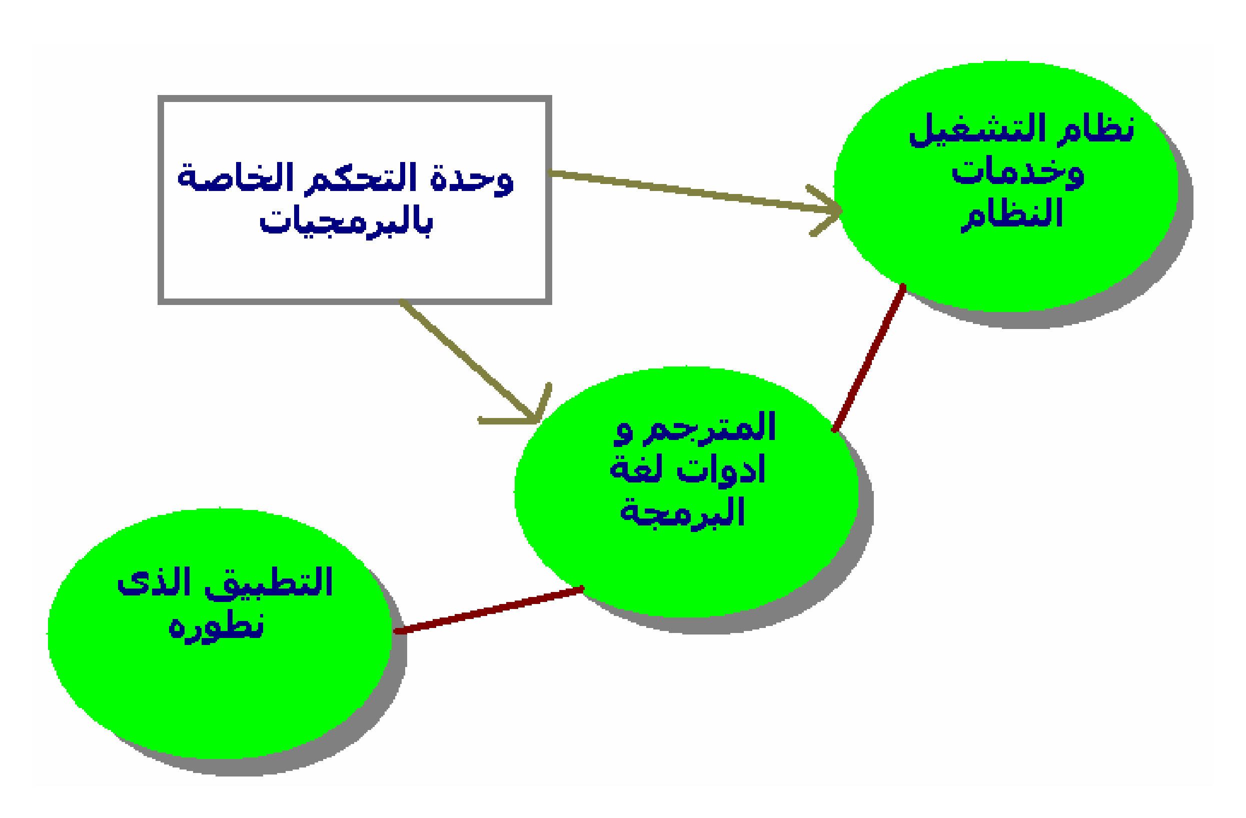
والسوال الان : ما هى وحدة التحكم الخاصة بالبرمجيات ؟

فـى الواقـع ان البرمجيـات التـى نقـوم بتطويرهـا باسـتخدام اللغـات الشـائعة عاليـة المسـتوى ليسـت برمجيات حقيقيـة بمعنـى انها ليسـت بـرامج Programs - وانما تطبيقات Applications ان البـرامج هـى التـى تملـك السـيطرة التامة علـى الحاسـب

وتتعامل معه مباشرة بينما التطبيقات لا تملك ذلك لانها تعتمد على وحدات تحكم اخرى مثل تطبيقات Windows

التى تعتمد على هذا النظام فى كل شى ولهذا يمكن لنظام Windows ان يتحكم فى عمل التطبيقات ويوقف عملها بل ويمنع عنها بعض الخدمات عند الحاجة لانه اصبح وسيط بين التطبيقات وبين العتاد – او الحاسب وهذا مايعرف بوحدة تحكم الخاصة بالبرمجيات من المستوى الاول.

وهناك مايعرف بوحدة التحكم الخاصة بالبرمجيات من المستوى الثانى وهى نتيجة مترجم لغة البرمجة والذى قد يضيف خواص للتطبيقات التى تقوم بانشائها وان لا تعلم عنها شيئا اما وحدة التحكم الخاصة بالبرمجيات من المستوى الثالث فهى نتيجة الخدمات التى يطلبها التطبيق من تطبيق اخر والتى قد ينتج عنها ردود غير مرغوب فيها ومن السهل تجنب ذلك ولهذا فان هذا النوع من وحدات التحكم لا يندرج تحت دراستنا.



شكل (٢) : يوضح عناصر وحدة التحكم الخاصة بالبرمجيات

وبفهم هذا المفهوم نصل الى ان

" من الممكن لنظام التشغيل ان يتحكم باسلوب تنفيذ التطبيق بقواعد معينة وبالمثل لغة البرمجة وقد يصل هذا التحكم لمستويات لا تخدم تنفيذ التطبيق مثل منعه من اداء عمليات معينة او اغلاقه فى اى وقت "

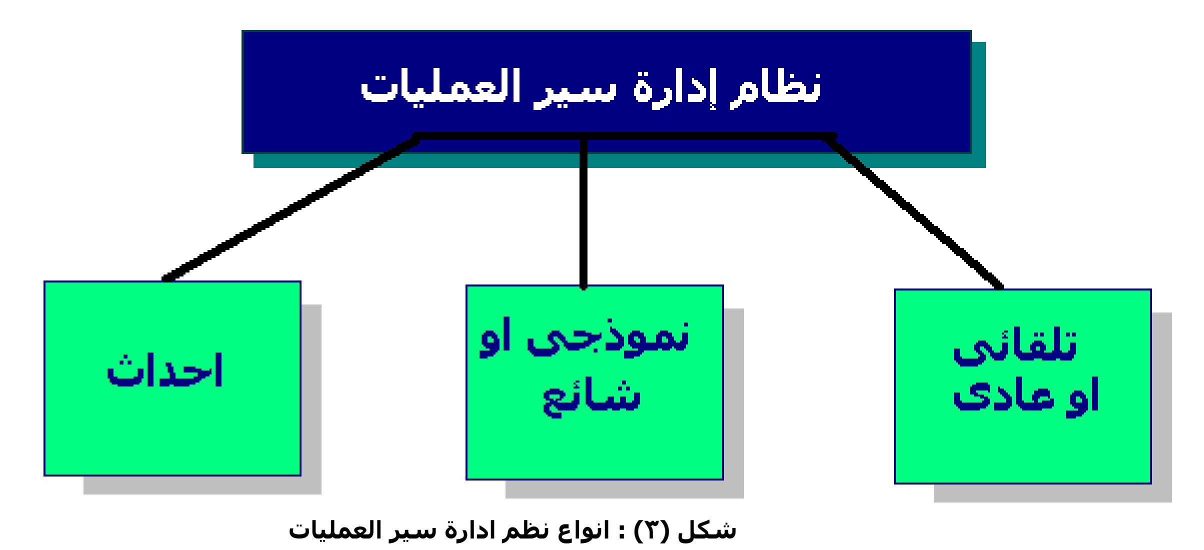
س : عفوا – كيف تتحكم لغة البرمجة بالتطبيق بعد ترجمته واستخراج ملف تنفيذى للعمل ؟

جـ : تتحكم لغة البرمجة باضافة اكود الى تطبيقك وترجمتها معه وانت لا تعلم شى عنها.
ولهذا قد تكون مبرمج تطبيقات محترف ومع ذلك يفشل تطبيقك – وينهار اثناء العمل نتيجة خطا فى نظام التشغيل – او خطا فى لغة البرمجة ! – وهذا ليس نادرا كما يعتقد البعض ولكنه لا يحدث الا مع المبرمجين المحترفين الذين يطورون تطبيقات مميزة والا لما سمعنا عن Service Pack وغيرها من Upgrade و Software Patches

لماذا ندرس ذلك ؟
جـ : فى الواقع هناك فائدتين
١ – مراعاة دراسة نظام التشغيل جيدا ولغة البرمجة عند تطوير التطبيقات (ليس هدفنا الاساسى فى هذا الدراسة)
٢ – سوف تقوم انت شخصيا بتطوير وحدة التحكم الخاصة بالبرمجيات مما يعنى انك سوف تتدخل فى عمل نظام التشغيل ولغة البرمجة من اجل السيطرة على التطبيقات او البرامج او النظم التى تقوم بعملها.

نظام إدارة سير العمليات

ان عملية ادارة سير العمليات تخضع لاحد الاحتمالات الثلاثة التالية كما نرى بالشكل ٣



شكل (٣) : انواع نظم ادارة سير العمليات

١ – تلقائى او عادى

ويقصد به عدم التدخل اساسا فى نظام سير العمليات – اى استخدام النظام المتوفر بصرف النظر عن نوعه – اى كما كنا نفعل عند تطوير انظمة قواعد البيانات فنحن لا نهتم على الاطلاق بنظام سير العمليات فمثلا عندما كنا نعمل تحت نظام دوس Dos داخل اللغة الشهيرة كليبر Clipper كنا نستعمل النظام النموذج Modal Model والذى تعتمد عليه تطبيقات اللغة – بينما عندما نعمل تحت Windows لتطوير تطبيقات

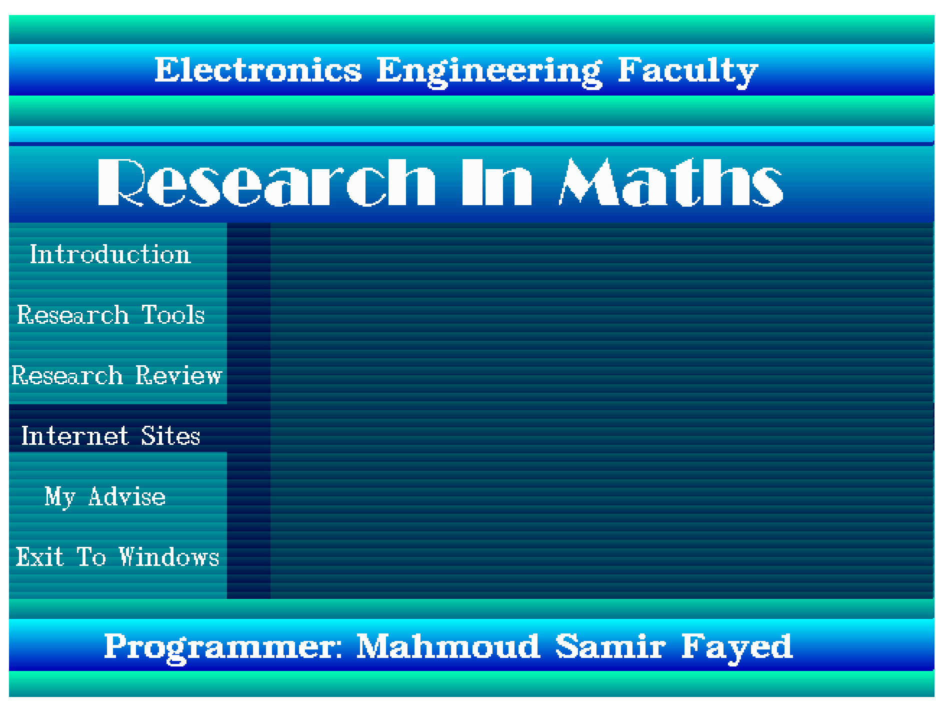
رسومية GUI Applications فاننا نعتمد على نظام الاحداث Events Model والتى يهتم بها نظام Windows ويتابعها ثم تستقبلها التطبيقات لتتعامل معها.

اى عندما نقول ان نظام سير العمليات تلقائى او عادى فاننا نقصد بذلك استبعاد تاثير عنصر ادارة سير العمليات على السوفت وير الذى نطوره.

٢ – نموذجى (شائع)

سبق وان اشرنا انه فى هذا النظام يتوقف البرنامج لاستقبال حدث محدد بعينه من المستخدم كاختيار عنصر من قائمة – او ادخال بيانات وهكذا واثناء ذلك الانتظار لا يعمل البرنامج اى شى – والجدير بالذكر هنا ان مكونات مثل هذا النظام يسهل برمجتها لانها لا تتداخل معا اثناء العمل وتمتلك كل وحدة من مكونات النظام السيطرة على وحدات الادخال والاخراج بدون تداخل مع المكونات الاخرى للنظام.

وسوف ناخذ مثال على ذلك شكل (٤) وشكل (٥) وهى تطبيقات قديمةسبق لى وان قمت بعملها داخل نظام التشغيل Dos وهى تعمل ايضا تحت Windows بدون اى مشاكل



شكل (٤) : برنامج تحت نظام DOS القديم يعمل بنظام Modal Model



شكل (٥) : برنامج قديم تحت Dos يعمل بنظام Modal Model

فى شكل (٤) نجد ان البرنامج فى حالة انتظار حتى يختار المستخدم اختيار محدد من ٦ خيارات مقدمة ويتم الاختيار فقط باستخدام لوحة المفاتيح – وهنا لايوجد اساسا فرصة للمستخدم لعمل اى شى اخر كما لا توجد فرصة للبرنامج لعمل مهام فى الخلفية لان نظام سير العمليات المتبع لا يسمح بذلك

فى شكل (٥) نفس الفكرة ولكن هناك دعم لاستخدام اى من الفارة او لوحة المفاتيح حتى يتم اختيار اى عنصر فى القائمة المقدمة والتى تشمل ٥ اختيارات.

ونلاحظ ان هذه البرامج رسومية وتعمل فى نمط 256 color ولها Skin خاص بها وهذا لم يكن شائعا فى تطوير تطبيقات Dos التى غالبا مع تكون Text Mode.

٣ – الاحداث

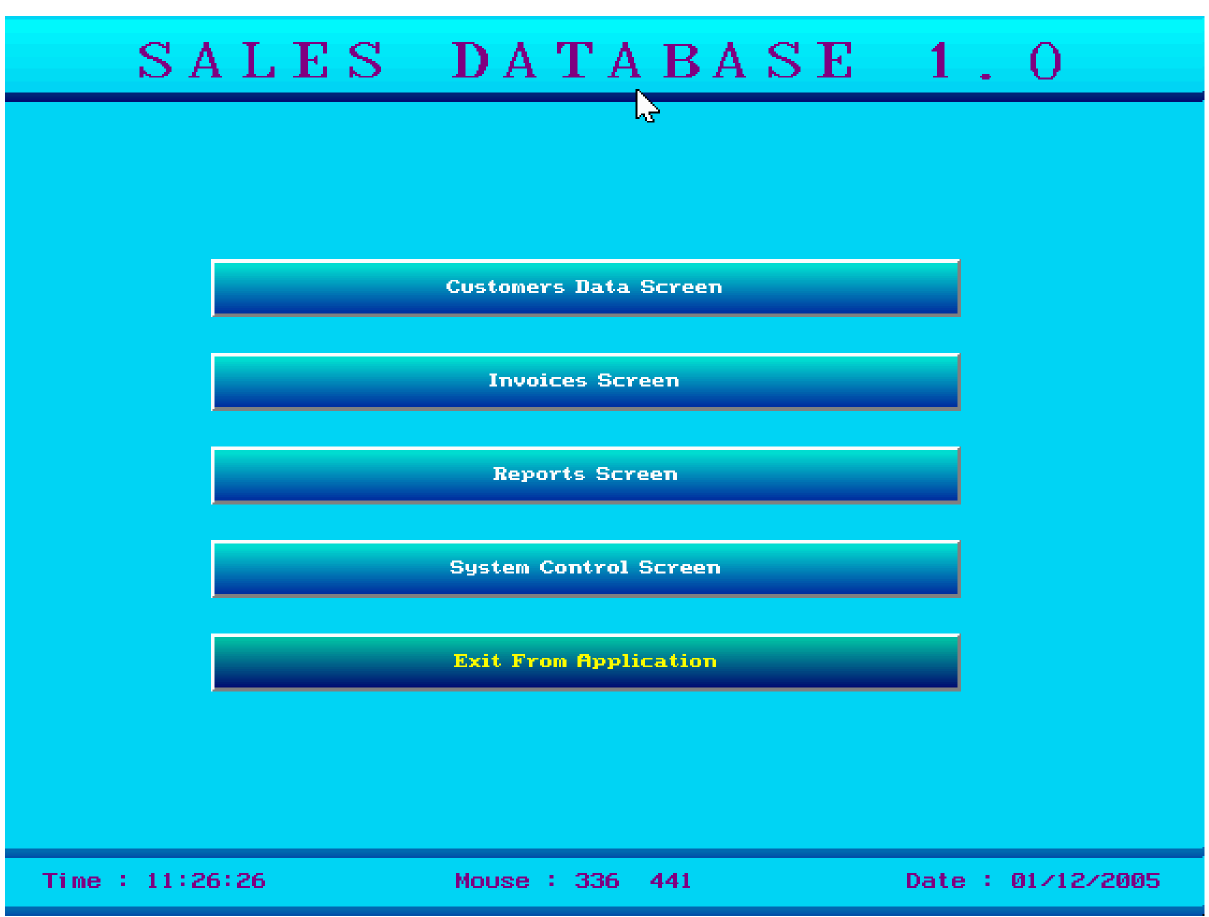
هنا يظل النظام فى حلقة عمل مستمرة منتظرا حدوث اى حدث يحدده المستخدم عن طريق وحدات الادخال مثل لوحة المفاتيح والفارة وفى نفس الوقت لا يتوقف النظام على ذلك بل يمكنه عمل مهام اخرى Background tasks كما يمكنه تنفيذ احداث كل فترة زمنية محددة Timer

انظر شكل (٦) و(٧) والتى تعرض احد البرامجيات التى سبق لى وان طورتها تحت Dos وتعمل بهذا الاسلوب

ونلاحظ امكانية العمل بكل من الفارة ولوحة المفاتيح معا فى نفس الوقت كما يعرض البرنامج تغير الوقت باستمرار – حيث ان البرنامج مدعم بنظام ادارة احداث (نظام ادارة سير العمليات باسلوب الاحداث)

وفى شكل (٧) نجد ان البرنامج يسمح باختيار اى مربع نص حتى يتم ادخال البيانات به كما يتيح فى اى وقت عمل Click لاى من الازرار Command Buttons لحفظ البيانات او الغاء عملية الحفظ .

بعد هذه الجولة سوف نتعرض الان الى مجموعة من المفاهيم التى تربط ما رايناه بما ينبغى ان نستوعبه – وخاصة ان هناك الكثير من المفاهيم المتداخلة والتى ينبغى توضيح كل منها حتى لا تختلط الامور



شكل(٦) برنامج يعمل تحت Dos بنظام Event Model

شكل (٧) -: شاشة ادخال بيانات مدعمة بنظام ادارة الاحداث – تعمل تحت Dos

مفاهيم اساسية -:

- نموذج سير العمليات لا يتم تحديده دائما بسهولة من خلال واجهة البرنامج
- قد تشتمل واجهة البرنامج على امكانيات توحى بان نظام سير العمليات المستخدم هو نظام الاحداث ويكون الواقع غير ذلك
- قد يكون نظام سير العمليات المستخدم هو نظام ادارة الاحداث ولا تكون الواجهة تتناسب معه – فتظن ان النظام المستخدم هو Modal
- ان عمليات التنقل بين عناصر واجهة البرنامج المختلفة مثل مربعات الادخال وازرار الاوامر تدخل تحت نظام البورة الخاص بنظام واجهة البرنامج ولا تشترط ان يكون النظام المتبع لادارة سير العمليات هو نظام الاحداث
- البرهان الاساسى لوجود نظام ادارة سير العمليات – باسلوب الاحداث هو تنفيذ مجموعة من التعليمات (الحدث) تلقائيا بدون تدخل من المستخدم بجرد تحقق شرط معين مع امكانية تنفيذ مجموعة من التعليمات بصفة دوريةTimer
- وجود Timer داخل البرنامج ليس برهان على وجود نظام ادارة سير العمليات

- قد توجد تطبيقات تعمل باسلوب الاحداث – ومع ذلك لا تشتمل على نظام لادارة الاحداث وانما يقوم المبرمج ببرمجة كل مجموعة من الاحداث يدويا
- نموذج سير العمليات لا يرتبط هل الواجهة Text base او Graphic فقد يكون نظام الاحداث Model ويعمل فى بيئة رسومية (غير شائع) وقد يكون نظام الاحداث Event ويعمل فى بيئة نصية (غير شائع)
- البيئة الرسومية لا تشترط ان تكون Graphic فقد تعمل فى Text mode وعندها تسمى Text Based GUI حيث يقصد بالـ GUI العناصر المكونة للواجهة مثل القوائم ومربعات الادخال والازرار ولا يقصد بها نمط الشاشة سواء كان نصى او رسومى.

ملحوظة هامة :-

تذكر جيدا ان سوف ندرس كيفية تصميم وبرمجة نظام لادارة سير العمليات يوفر فيما بعد تطوير تطبيقات مبنية عليه وتسخدم موارد النظام ويجب ايضا ان تعلم ان النظام الذى سوف نطوره يدخل ضمن وحدة التحكم الخاصة بالبرمجيات – نوعية لغات البرمجة لاننا نطور اداة نستخمها مع لغة البرمجة لتطوير النظم – واذا كان النظام الذى نطوره هو اساسا عبارة عن نظام تشغيل فان نظام سير العمليات الذى نطوره يدخل ضمن وحدة التحكم الخاصة بالبرمجيات – نوعية نظام التشغيل وللمراجعة انظر شكل (٢) الذى سبق عرضه.

وحدة التعليمات – الاكواد Code Block :-

يقصد بها مجموعة التعليمات المجمعة معا تحت اسم معين – على سبيل المثال اى دالة او وظيفة Function فى البرمجة الهيكلية او التركيبية Structure Programming يمكن اعتبارها Code Block وبالمثل كذلك اى Method فى برمجة الكائنات Object Oriented Programming (OOP) او اى مقاومة حدث Event فى نظام سير العمليات المبنى على الاحداث – او اى مقاومة Resistance فى برمجة الخادم الممتاز DoubleS (Super Server) Programming

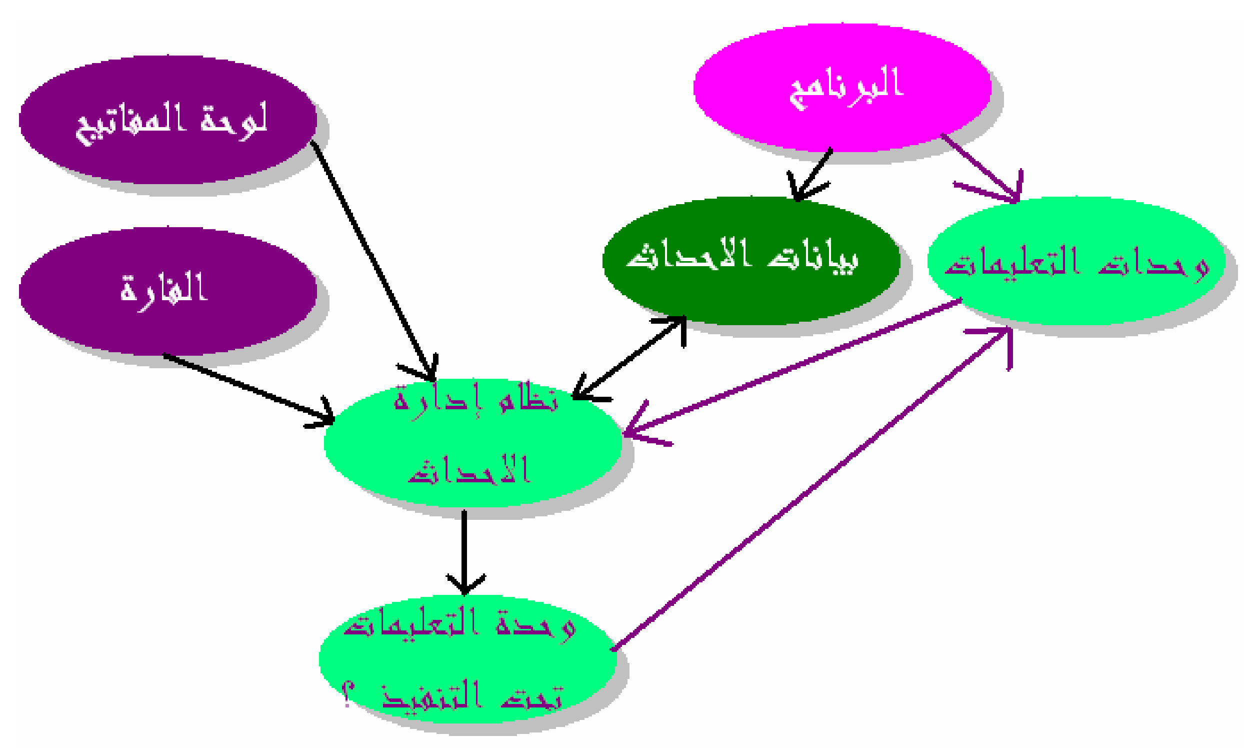
مهام نظام إدارة الاحداث Event-Driven System :-

- تسجيل بيانات وحدات التعليمات التى سوف يتم مناداتها باستمرار فى حلقة نظام ادارة الاحداث
- توفير امكانية المتابعة اللحظية لنظام سير العمليات – لمعرفة اى وحدة تعليمات يجرى تنفيذها
- امكانية ايقاف النظام عن العمل فى اى وقت
- امكانية ادخال وحدات تعليمات جديدة الى حلقة النظام اثناء العمل
- امكانية حذف وحدات تعليمات من حلقة النظام اثناء العمل
- امكانية المتابعة التلقائية لوحدات الادخال المختلفة كلوحة المفاتيح والفارة وارتباط احداث بها

كيفية إستخدام نظام ادارة الاحداث :-

يتم تعريف وحدات التعليمات الاساسية ثم بعد ذلك نطلب من نظام ادارة الاحداث بدء العمل وعندها نفقد السيطرة Control على سير العمليات والتى يمتلكها بدوره نظام ادارة الاحداث ومع ذلك لا نفقد السيطرة على سير النظام لانه يكون مرن بالحد الذى يسمح بعمل اى شىء.

ميكانيكية عمل نظام إدارة الاحداث :-



شكل (٨) : ميكانيكية عمل نظام ادارة الاحداث

يبدا البرنامج بتعريف كل من وحدات التعليمات والتى تكون Functions او Methods ثم بعد ذلك يحدد لنظام سير العمليات اى الوحدات يتم تنفيذها داخل حلقة النظام فى البداية وذلك من خلال "بيانات الاحداث" ثم بعد ذلك ينتقل التجكم الى نظام ادارة الاحداث الذى بدوره يعمل على تنفيذ وحدات التعليمات بالتتابع (واحدة تلو الاخر) من خلال بيانات الاحداث وفى نفس الوقت يستقبل بيانات وحدات الادخال لتكون جاهزة للاستخدام من قبل الاحداث.

س : كيف تكون احداث وهى يتم تنفيذها بالتتابع من قبل نظام ادارة الاحداث ؟

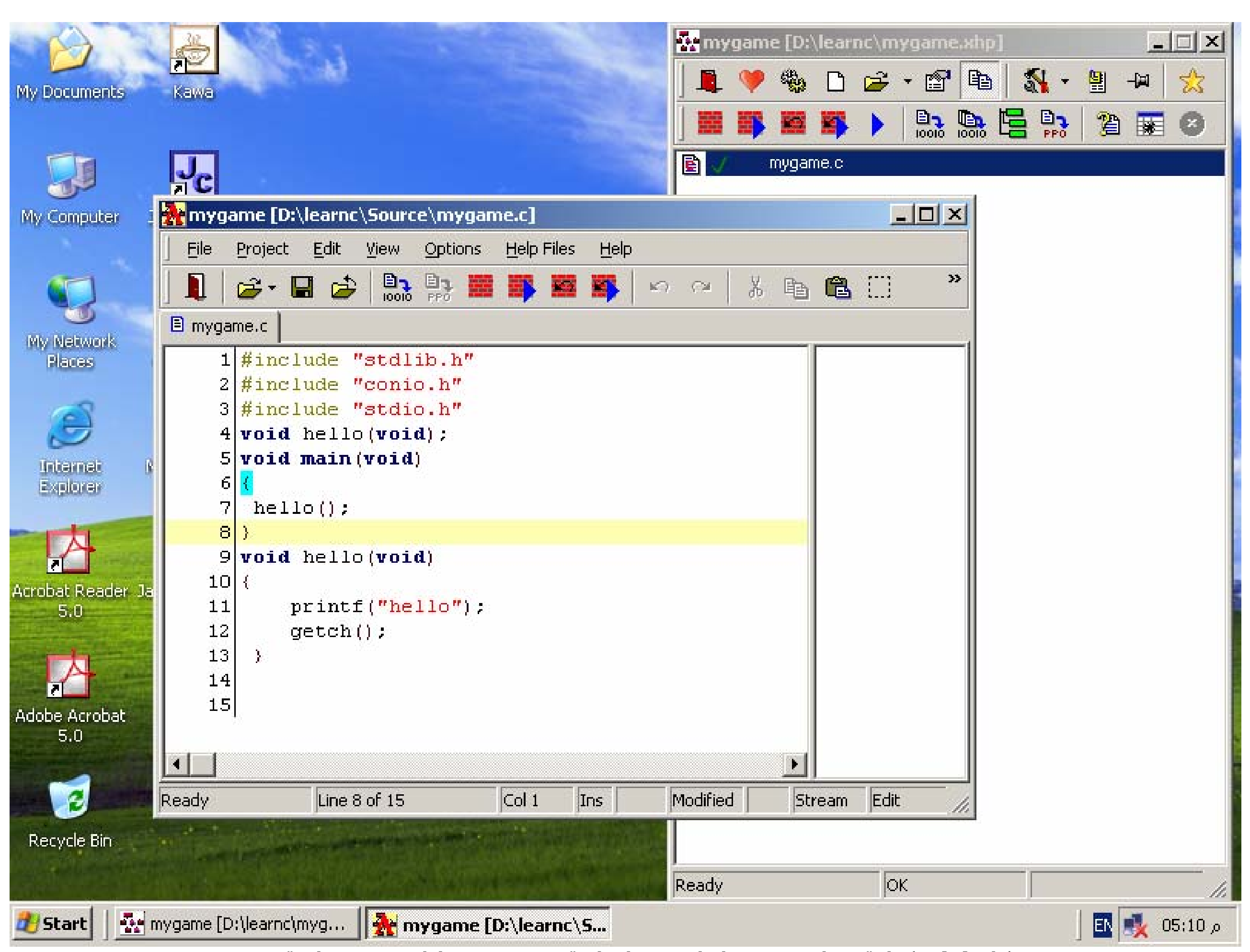
جـ : احسنت – فالسوال يدل على فهمك للحدث وهو الذى يرتبط بشرط معين لحدوثه – الاحداث هنا تتحقق دائما ويتم تنفيذها مباشرة بالتتابع وهى هنا تسمى Default Events ولعمل حدث بشرط معين يكفى ان تبدا وحدة التعليمات بصرف النظر كانت

Function او Method او Resistance – يمكنك ان تبداها بجمل if لكى تختبر تحقق الشرط Condition الخاص بالحدث قبل تنفيذه.

اى ان الحدث هو عبارة عن وحدة تعليمات تبدا بجملة if statement ويتم مناداتها باستمرار داخل حلقة نظام ادارة الاحداث.

مفهوم موشر وحدة التعليمات Code Block Pointer:-

اذا كنت تتذكر اساسيات البرمجة بلغة سى C language فان تعلم جيدا انه بمجرد عمل prototype لاى Function فانه يمكن استخدامها مباشرة عن طريق كتابة اسم Function يليها قوسين متقابلين هكذا (). والمثال التالى يوضح كيفية مناداة الدالة hello().



شكل (٩) : كتابة برنامج بسيط باستخدام لغة سى من خلال IDE مجانية تسمى xMate

#include "stdio.h"

```
#include "conio.h"
#include "stdlib.h"

Void hello(void);

Void main(void)
{
   hello() ;
}

void hello(void)
{
   printf("Welcome to my C program ! ");
   getch();
}
```

شكل (٩) السابق يوضح كيف تم كتابة البرنامج من خلال الاداة xMate وهى عبارة عن IDE تدعم العديد من لغات البرمجة مثل C,xBase & xHarbour ويمكن الحصول عليها مجانا من خلال الانترنت.

والسوال الان :-

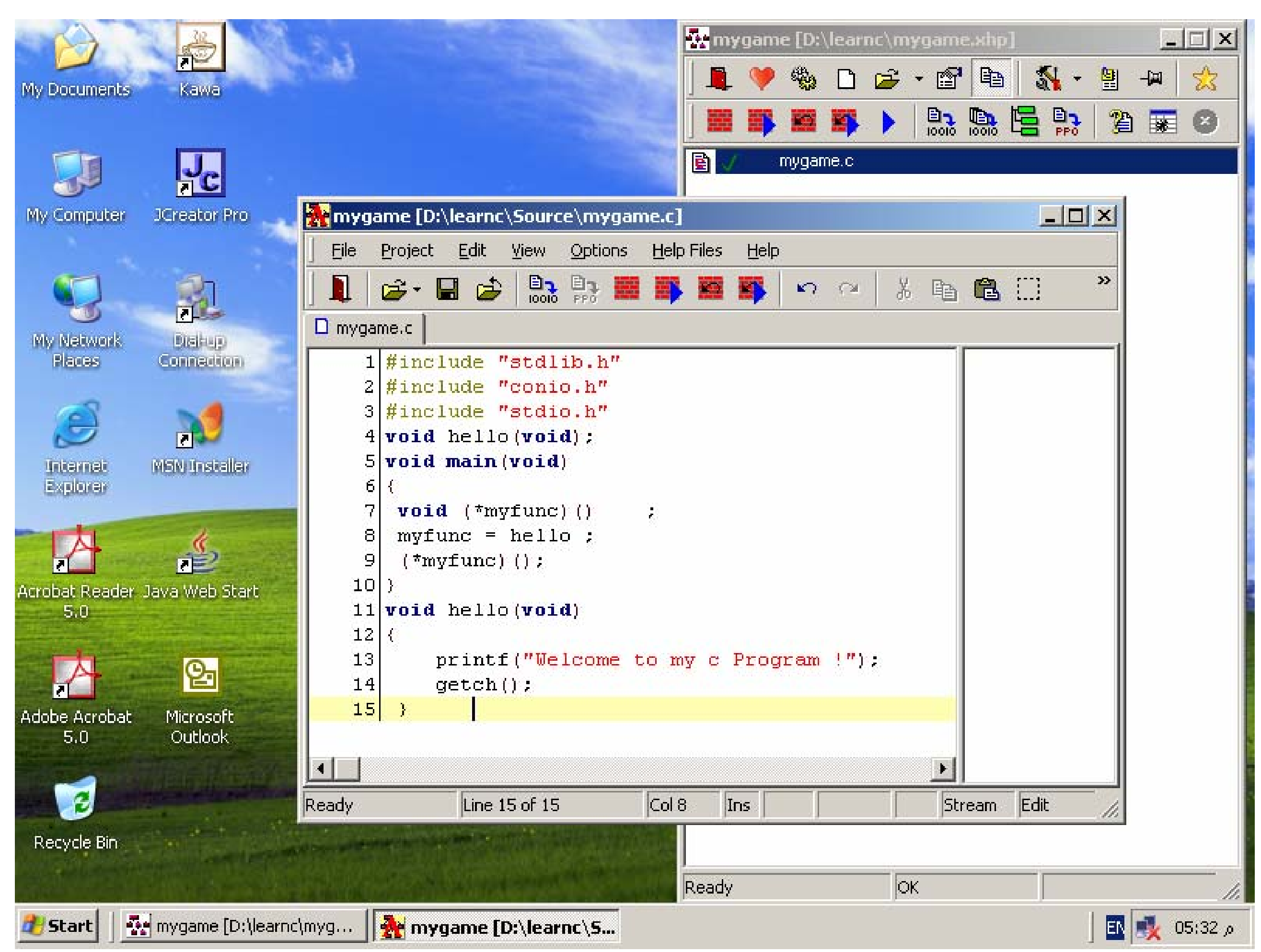
س : كيف يمكن مناداة الدالة ()hello من قبل نظام ادارة الاحداث ؟

جـ : – لا يمكن لنظام ادارة الاحداث مناداة الدالة مباشرة من خلال اسمها – لانه يفترض ان لا يعرف نظام ادارة الاحداث اسم الدوال التى يقوم بمناداتها مباشرة – وانما يستقبل معلومات تشير اليها من خلال بيانات الاحداث اى بمعنى اوضح يكون اسم الدالة داخل متغير يحتفظ به نظام ادارة الاحداث – وعند الحاجة يقوم نظام ادارة الاحداث باستخراج اسم الدالة من المتغير ثم يقوم بمناداتها – فكيف يتم برمجة ذلك – وما اسم هذه العملية ؟
اسم هذه العملية هو Calling Function by reference/pointer اى مناداة الدالة عن طريق موشر ولاننا هنا فى نظام ادارة سير العمليات فاننا نسمى ذلك مناداة وحدة التعليمات عن طريق موشر لان وحدة التعليمات ممكن ان تكون Function او Method وهكذا.

والمثال التالى يوضح كيفية استدعاء دالة عن طريق موشر :-

```
#include "stdlib.h"
#include "conio.h"
#include "stdio.h"
void hello(void);
```

```
void main(void)
{
 void (*myfunc)()    ;
 myfunc = hello ;
 (*myfunc)();
}
void hello(void)
{
    printf("Welcome to my c Program !");
    getch();
 }
```



شكل (١٠) : استدعاء دالة عن طريق مؤشر باستخدام لغة سى.

ويمكنك ايضا عمل نفس الشى باستخدام لغات اخرى – مثل عائلة لغات xBase امثلة CA-Clipper, xHarbour & Visual FoxPro

فمثلا المثال التالى يوضح كيفية مناداة دالة فرعية

```
* TEST.PRG

Hello()
```

```
Function hello()
Set color to w/b
Clear
? "Welcome to my xBase Program"
inkey(0)
return
```

ولترجمة البرنامج باستخدام لغة كليبر CA-Clipper نستخدم المترجم Clipper.exe لاستخراج ملف Object ثم الرابط Linker وليكن Blinker على سبيل المثال لاستخراج ملف جاهز للتنفيذ (.exe) Executable File.

```
Clipper test
Blinker fi Test lib clipper,extend
```

ويمكنك ترجمة البرنامج ايضا باستخدام المترجم المجانى xHarbour وهو متنقل اى يدعم اكثر من نظام تشغيل مثل Dos,Windows,OS/2 & Linux وغيرها من الانظمة وتقوم فكرة هذا المترجم على تحويل الكود من xBase الى لغة C ويقدم مكتبات لتستخدم فى عملية الربط.

ونظرا لان المنصة الشائعة لتطوير التطبيقات هى Microsoft Windows فانى استخدم xHarbour/MiniGUI والتى توفر امكانية تطوير GUI WIN32 Applications ويمكن الحصول عليها مجانا من الموقع
http://www.sourceforge.net/projects/harbourminigui
او من خلال الموقع http://www.geocities.com/harbour_minigui

وعند تحميلها على الجهاز سوف تجد ملف compile.bat يستخدم فى ترجمة البرامج مباشرة ومن المفترض ان يتم تحميلها على المسار C:\HMG وسوف يكون مسار الملف المستخدم فى الترجمة C:\HMG\BATCH\COMPILE.BAT
وتوفر عليك هذه الحزمة تحميل xHarbour منفردا او مترجم لغة سى – لذا انت لست بحاجة لاى شى غيرها كما انها حزمة مجانية مئة بالمئة.

وفى المثال الذى نقف عنده تتم عملية الترجمة ببساطة كالتالى

```
Compile test
```

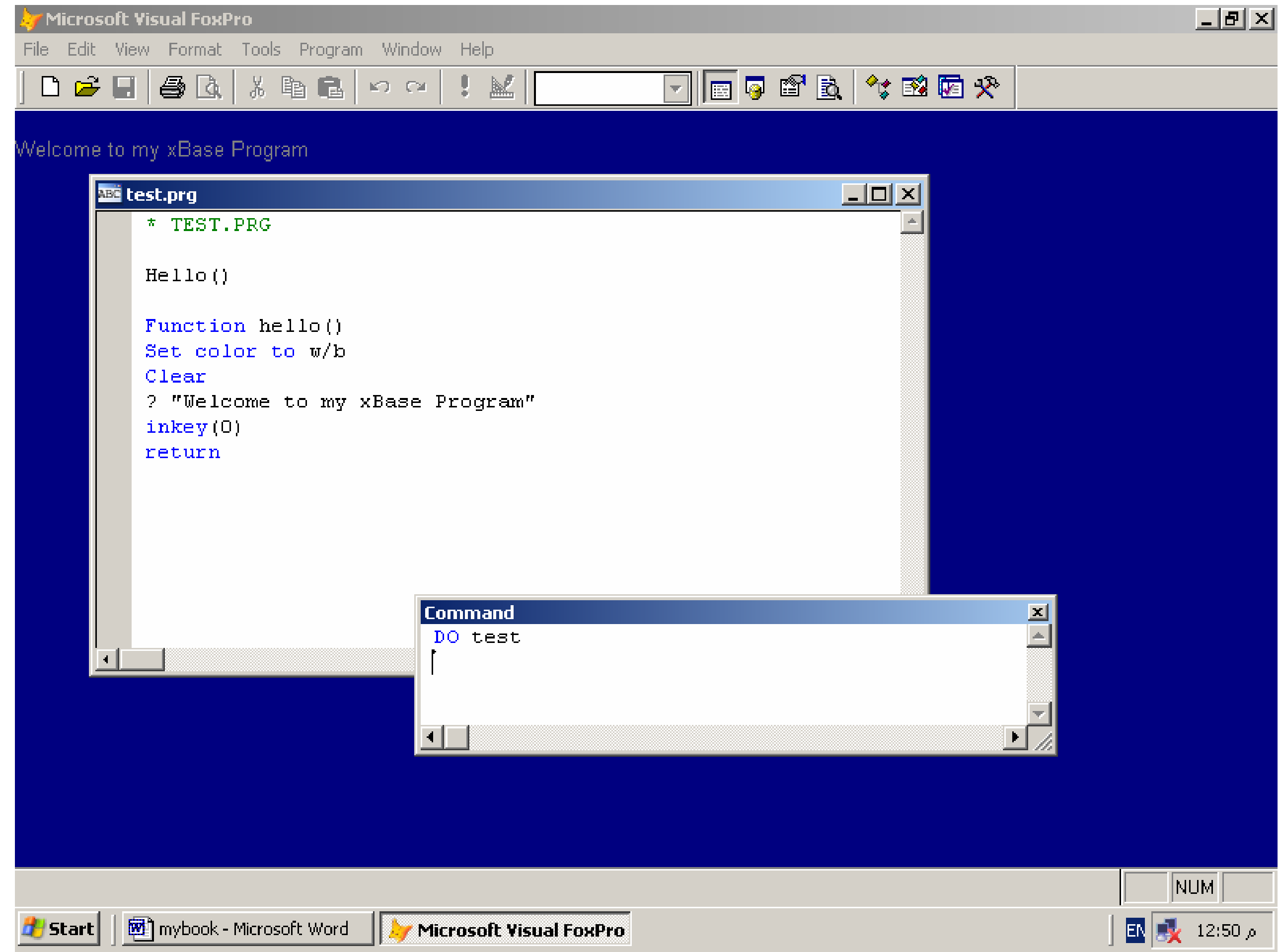
وسوف يكون البرنامج الذى تم انتاجة Win32 console application مع معرفة ان xHarbour/MiniGUI تتيح عمل GUI تحت Windows بسهولة ويسر.

وبالتاكيد يمكننا ترجمة البرنامج باستخدام لغة ++xBase وهى من انتاج شركة Alaska الالمانية والتى نستخدم فيها المترجم xpp.exe لاستخراج ملف Object والرابط alink لاستخراج ملف تنفيذى executable

```
xpp test
alink test
```

والجدير بالذكر ان شركة Alaska قامت بانتاج Visual xBase++ وهى منصة تطوير مبنية على xBase++ وتتيح تطوير التطبيقات بسهولة.

وايضا يتاح لنا ترجمة البرنامج باستخدام لغة فيجوال فوكس برو Visual FoxPro من انتاج شركة Microsoft انظر شكل (١١) والذى يبين كيفية تنفيذ البرنامج من خلال نافذة الاوامر Command Window



شكل(١١) : تنفيذ البرنامج باستخدام فيجوال فوكس برو ٩ - Visual FoxPro 9

اما بخصوص المثال الثانى والذى ينادى الدالة عن طريق موشر فيتم بسهولة كالتالى

```
* TEST.PRG

LOCAL myfunc
myfunc = "Hello()"
&myfunc

Function hello()
Set color to w/b
Clear
? "Welcome to my xBase Program"
```

```
inkey(0)
return
```

وهنا فى هذا المثال ببساطة تم كتابة اسم الدالة فى متغير myfunc وعند الرغبة فى استدعائها تم استخدام العلامة & يليها اسم المتغير. ويمكن تنفيذ المثال باى لغة من لغات xBase.

الخلاصة :

- ان مناداة الدالة فى اى وقت من خلال موشر يتيح لنظام ادارة الاحداث مناداة الدوال عند الحاجة وذلك بعد تخزين موشرات لهذه الدوال داخل متغيرات او مصفوفة او اى هيكل بيانات Data Structure مناسب.

- لا ينبغى للمبرمج المحترف ان يتقيد بلغة برمجة او منصة تطوير بل ينبغى ان يستوعب المفهوم ومن ثم يمكنه ان يطبقه باى لغة سواء كانت لغة سى C او لغة من لغات xBase او غيرها من لغات البرمجة.

- من المفيد ان تستخدم لغات متنقلة متاحة فى اكثر من نظام تشغيل مثل C و xHarbour

هيكل بيانات النظام System Data Structure :-

لايوجد نظام لايحتاج لتخزين بيانات – بصورة موقته فى الذاكرة Ram – حيث تكون بيانات النظام شائعة بين اجزاءه والتى تتشارك فى استخدام البيانات.

هنا فى نظام ادارة الاحداث – لابد من بيانات تكون متاحة لاجزاء النظام وبالتاكيد العنصر الرئيسى لهذه البيانات هى البيانات اللازمة لتخزين معلومات عن وحدات التعليمات التى يتم مناداتها من قبل النظام باستمرار

انظر الى التعليمات التالية المكتوبة بلغة سى وتشكل خطوة للامام نحو اكتمال بناء نظام ادارة الاحداث

```
#include "stdlib.h"
#include "conio.h"
#include "stdio.h"

struct
{
  void (*myfunc)();
} myevents[50] ;

int myelement;
myelement = 0;

void addevent( void (*myfunc)() );
void doevents(void);

void hello(void);

void main(void)
{

 addevent(hello);
 addevent(getch);
 doevents();

}

void hello(void)
{
```

```
        printf("Welcome to my c Program !");
    }

    void addevent( void (*myfunc)() )
    {
      myelement++ ;
      myevents[myelement].myfunc = myfunc ;
    }

    void doevents(void)
    {
       int x ;
       for (x = 1 ; x < 50 ; x++)
         {
             if ( myevents[x].myfunc != myevents[0].myfunc  )
             {
                   (*myevents[x].myfunc)();
             }
             else
             {
                break ;
             }
       }
    }
```

فى هذا المثال نجد هيكل بيانات النظام مكون من Structure يسمى myevents يمكن اى يحتوى ٥٠ موشر ل ٥٠ دالة بالاضافة الى متغيرmyelement يحدد رقم اخر عنصر تم استخدامه من Structure والجدير بالذكر انه يمكن تسجيل ٤٩ موشر فقط لاننا استخدمنا الموشر رقم صفر zero للتميز هل الموشر يشير لدالة ام لا.

الدالة addevent تستخدم لاضافة دالة فى بيانات الاحداث بمساعدة المتغير myelement الذى يحدد رقم ال structure

الدالة doevents تقوم بتنفيذ الاحداث المسجلة مرة واحدة من خلال جملة For وتستخدم جملة if لتعرف هل ال structure يشير لدالة ام لا

الدالة main() تضيف للنظام الدالة hello() يليها getch() من خلال الدالة addevent() ثم تستدعى doevents() لكى تنفذ الدوال المسجلة.

تنظيم الشفيرة المصدرية :-

ان الكود فى المثال السابق يعد غاية فى البساطة لكنه فى الواقع من الناحية العلمية غير منظم ولعلك قد التمست ذلك اذا كنت من الذين قد اطلعوا على العديد من الشفيرات المصدرية المكتوبة بلغة سى

جميع الاكواد السابقة تم كتابتها فى ملف واحد – ولكن للتنظيم سوف يتم تقسيمها الى اكثر من ملف وبالتحديد كنقطة بداية سنقسم الكود فى المثال السابق الى ٤ ملفات

- SysData.h
  هذا الملف سوف يحتوى على هيكل بيانات النظام System Data Structure
- SysLib.C
  هذا الملف سوف يحتوى على الدوال الخاصة بالنظام مثل addevent() و doevents()
- SysUser.h
  هذا الملف سوف يحتوى على المعلومات الكافية لاستخدام دوال النظام
- Systest.C
  هذا الملف سوف يحتوى على برنامج اختبار النظام

SysData.h

```
#include "stdlib.h"
#include "conio.h"
#include "stdio.h"

struct
{
	void (*myfunc)();
} myevents[50] ;

int myelement;
myelement = 0;
```

SysLib.C

```
#include "sysdata.h"

void addevent( void (*myfunc)() )
{
        myelement++ ;
        myevents[myelement].myfunc = myfunc ;
}

void doevents(void)
{
         int x ;
         for (x = 1 ; x < 50 ; x++)
           {
                    if ( myevents[x].myfunc !=
      myevents[0].myfunc  )
                    {
                            (*myevents[x].myfunc)();
                    }
                    else
                    {
                      break ;
                    }
           }
}
```

SysUser.h

```
#include "syslib.c"

extern void addevent( void (*myfunc)() );
extern void doevents(void);
```

SysTest.c

```
#include "sysuser.h"

void hello(void);
void main(void)
{
```

```
        addevent(hello);
        addevent(getch);
        doevents();

}

void hello(void)
{
        printf("Welcome to my c Program !");
}
```

بالنسبة للغات xBase مثل فيجوال فوكس برو تكون التعليمات كالتالى

```
PUBLIC myevents[50]
PUBLIC myelement
myelement = 0

addevent("hello()")
mydoevents()
WAIT

FUNCTION hello()
    SET COLOR TO w/b
    CLEAR
    ? "hello"
RETURN

FUNCTION addevent(p1)
    myelement = myelement + 1
    myevents[myelement] = p1
RETURN

FUNCTION mydoevents()
    LOCAL x,r
    FOR x = 1 TO 50
         IF .not. EMPTY(myevents[x])
              r = myevents[x]
              r = &r
         ELSE
                 EXIT
         ENDIF
    ENDFOR
RETURN
```

وبالتاكيد يمكن تنظيم الشفيرة المصدرية – لتلائم تطور النظام بعد ذلك
سوف نقسم الكود الى ملفين

- SYSLIB.PRG
  سوف يحتوى على هيكل بيانات ودوال او وظائف النظام
- SYSTEST.PRG
  سوف يحتوى على برنامج لاختبار النظام

SysLib.PRG

```
FUNCTION SYS_START()
PUBLIC myevents[50]
PUBLIC myelement
myelement = 0
RETURN

FUNCTION addevent(p1)
    myelement = myelement + 1
    myevents[myelement] = p1
RETURN

FUNCTION mydoevents()
    LOCAL x,r
    FOR x = 1 TO 50
        IF .not. EMPTY(myevents[x])
                        r = myevents[x]
                        r = &r
        ELSE
                EXIT
        ENDIF
    ENDFOR
RETURN
```

SysTest.PRG

```
SET PROCEDURE TO syslib.prg ADDITIVE
SYS_START()
addevent("hello()")
mydoevents()
WAIT
```

```
FUNCTION hello()
    SET COLOR TO w/b
    CLEAR
    ? "hello"
RETURN
```

مفهوم الة الانتظار State Machine :-

يبقى لدينا خاصية هامة ينبغى برمجتها فى نظام ادارة الاحداث الخاص بنا – هذه الخاصية هى امتلاك نظام ادارة الاحداث للتحكم الكامل بالنظام وذلك من خلال مفهوم ال State Machine ويقصد بها ان يحتوى النظام While Loop تعمل للابد حيث يكون الشرط الخاص بها دائما True – فى الواقع كل ما علينا فعله هو ان الدالة او الوظيفة DOEVENTS() المسئولة عن تنفيذ الاحداث لابد ان تمتلك التحكم Control من خلال مفهوم State Machine

يبقى لنا شى هام و هو اضافة متغير لهيكل بيانات النظام System Data Structure يمكننا من خلاله ايقاف النظام عن العمل عند تحقق شرط معين وليكن مثلا عند ضغط اى مفتاح من لوحة المفاتيح على سبيل المثال.

SysData.h

```
#include "stdlib.h"
#include "conio.h"
#include "stdio.h"

struct
{
  void (*myfunc)();
} myevents[50] ;

int myelement;
myelement = 0;

int shutdown;
shutdown = 0;
```

SysLib.C

```
#include "sysdata.h"

void addevent( void (*myfunc)() )
{
          myelement++ ;
          myevents[myelement].myfunc = myfunc ;
}

void doevents(void)
{
         int x ;

         do {
             if (shutdown == 1)
                break;
             for (x = 1 ; x < 50 ; x++)
               {
                  if ( myevents[x].myfunc != myevents[0].myfunc  )
                  {
                           (*myevents[x].myfunc)();
                  }
                  else
                  {
                   break;
                  }
               }
         } while(1);
}
```

SysTest.c

```
#include "sysuser.h"

void hello(void);
void mybreak(void);
```

```
void main(void)
{

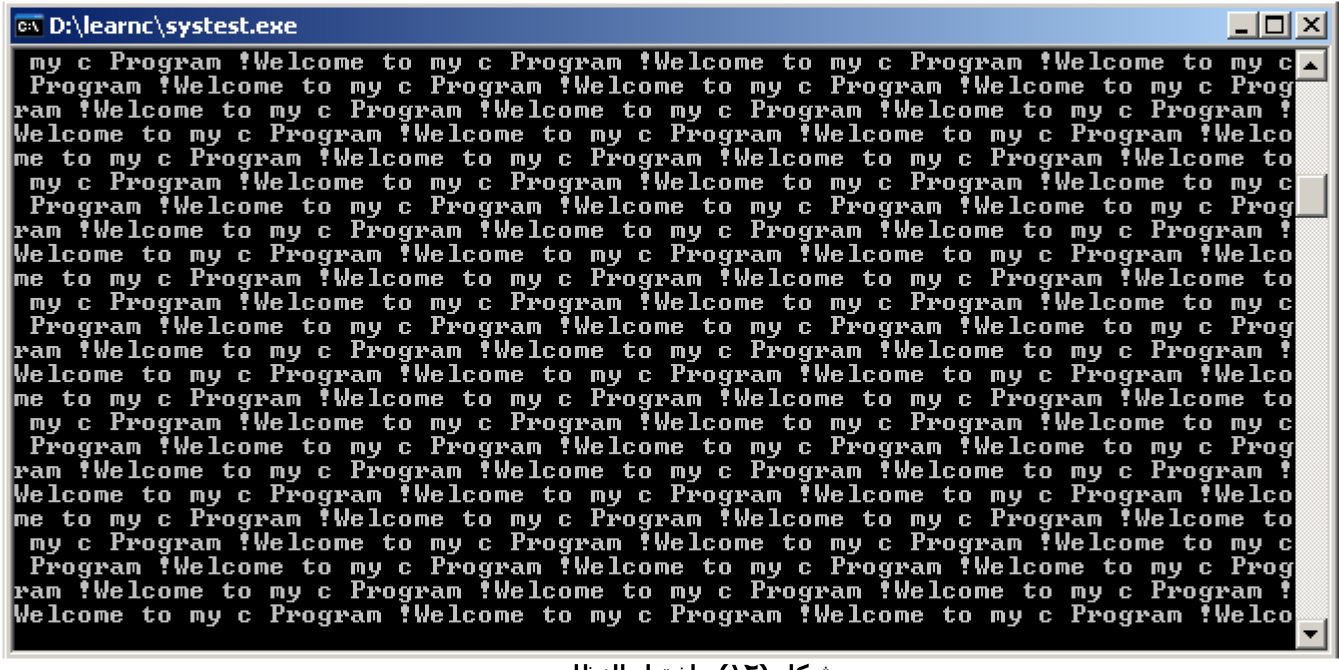
 addevent(hello);
 addevent(mybreak);
 doevents();

}

void hello(void)
{
    printf("Welcome to my c Program !");
}

void mybreak(void)
{
   if (kbhit())
      shutdown = 1 ;
}
```

والشكل التالى رقم (١٢) يبين نتيجة تنفيذ البرنامج – المسئول عن اختبار النظام حيث يتم عرض العبارة النصية "Welcome to my c program" باستمرار حتى يتم ضغط اى مفتاح من لوحة المفاتيح وعندها يتم انهاء النظام.



شكل (١٢) . اختبار النظام

ونلاحظ من التعديلات التى لحقت بالكود – ان الملف SysUser.h لم يظهر به اى تعديل

كما نلاحظ ان النظام الان يستجيب لتغير قيمة المتغير Shutdown حيث عندما تكون قيمة هذا المتغير تساوى واحد فانه يتم اغلاق النظام ولهذا فى الملف SysTest.C تم اضافة الدالة او الوظيغة ()mybreak والتى مسئولة عن اغلاق النظام من خلال المتغير Shutdown بمجرد ضغط اى مفتاح من خلال لوحة المفاتيح – وتم الاستدلال على ضغط اى مفتاح من خلال الدالة ()kbhit.

وفيما يلى عرض لنفس النظام باستخدام لغات xBase

SysLib.PRG

```
FUNCTION SYS_START()
PUBLIC myevents[50]
PUBLIC myelement
PUBLIC sysshutdown
sysshutdown = .f.
myelement = 0
RETURN


FUNCTION addevent(p1)
    myelement = myelement + 1
    myevents[myelement] = p1
RETURN

FUNCTION mydoevents()
LOCAL x,r
DO WHILE .t.
    FOR x = 1 TO 50
         IF .not. EMPTY(myevents[x])
             r = myevents[x]
             r = &r
         ELSE
            EXIT
         ENDIF
    ENDFOR
    IF sysshutdown = .t.
        EXIT
    ENDIF
ENDDO
RETURN
```

SysTest.PRG

```
SET PROCEDURE TO syslib.prg ADDITIVE
SYS_START()
addevent("hello()")
addevent("mybreak()")
mydoevents()
WAIT

FUNCTION hello()
    SET COLOR TO w/b
    CLEAR
    ? "hello"
RETURN

FUNCTION mybreak()
    i = INKEY()
    IF i != 0
    sysshutdown = .t.
ENDIF
RETURN
```

ملحوظة هامة :-

عند تطبيق النظام باستخدام احد لغات xBase التى تعمل تحت Microsoft Windows مثل فيجوال فوكس برو VFP او xHarbour/MiniGUI فانك سوف تواجه مشكلة بسيطة وهى تعطل استجابة برنامجك لاحداث الواجهة User Interface Events الرسومية GUI والحل هو ان تشتمل الدالة ()mydoevents على امر يستدعى احداث النظام (نظام ادارة الاحداث الخاص بنظام التشغيل) وهذا الامر هو DO EVENTS

كيفية تصميم نظام ادارة الاحداث :-

لقد تعرضنا سابقا للمفاهيم الاساسية الازمة لوضع حجر الاساس لنظام ادارة الاحداث – ومن هنا يمكنك الانطلاق لتصميم نظام ادارة الاحداث الخاص بك لتضيف اليه المميزات التى ترغب بها لتلائم حاجتك تبعا لنوعية التطبيقات او النظم التى تقوم بتطويرها

والسوال الان – كيف يختلف تصميم نظام ادارة الاحداث من شخص لاخر ؟

ان نقاط الاختلاف تكمن فى :-

- الامكانيات التى يوفرها النظام مثل معرفة حالة النظام اثناء العمل وامكانية التحكم بمجموعة من الاحداث معا وهكذا
- هيكل البيانات يختلف تبعا لقدرات النظام وطريقة العمل المتبعة
- نمط البرمجة فمثلا قد يكون Structure Programming مثل الامثلة السابقة وقد يكون Object Oriented

مثال :- تصميم لنظام ادارة احداث يتلائم مع اغلب متطلبات العصر وهو من ابتكار المؤلف.

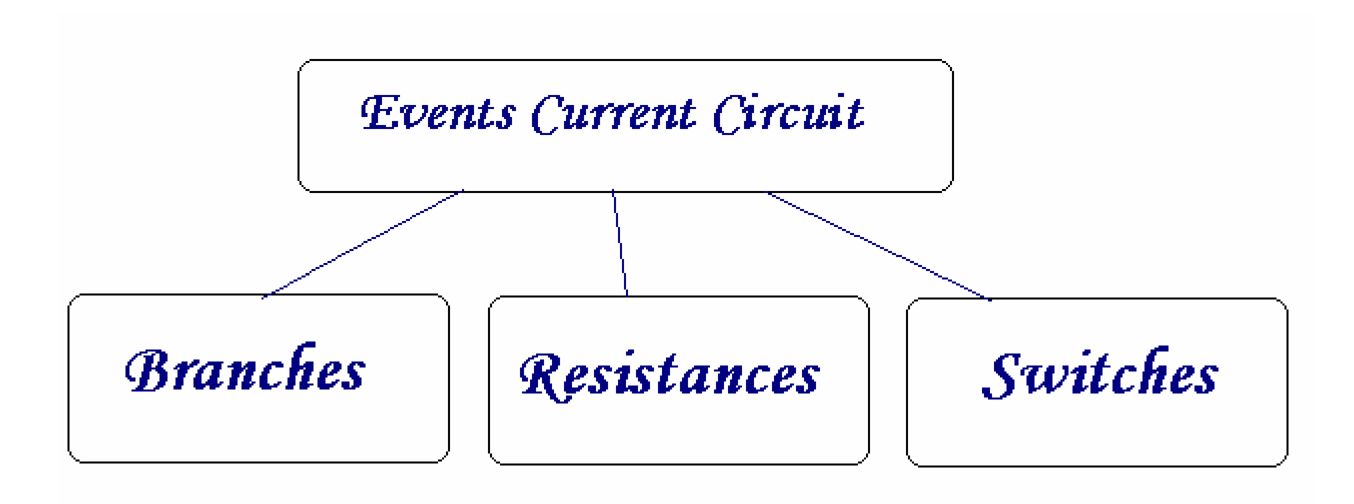
١ - اسم النظام : دائرة تيار الاحداث Events Current Circuit
٢ - هيكل البيانات : عبارة عن محاكاة Simulation للدوائر الكهربية
٣ - نمط البرمجة : برمجة الكائنات Object Oriented Programming

الامكانيات التى يوفرها النظام :-

- التحكم فى الاحداث من خلال تقسمها الى مجموعات(دوائر Circuits) ومجموعات فرعية (فروع Branches)
- امكانية معرفة حالة النظام فى اى وقت (تحديد نقطة العمل اللحظية للنظام)
- مرونة النظام لاستقبال حدث او مجموعة من الاحداث الجديدة اثناء العمل
- امكانية حذف حدث او مجموعة من الاحداث من النظام اثناء العمل

مميزات النظام :-

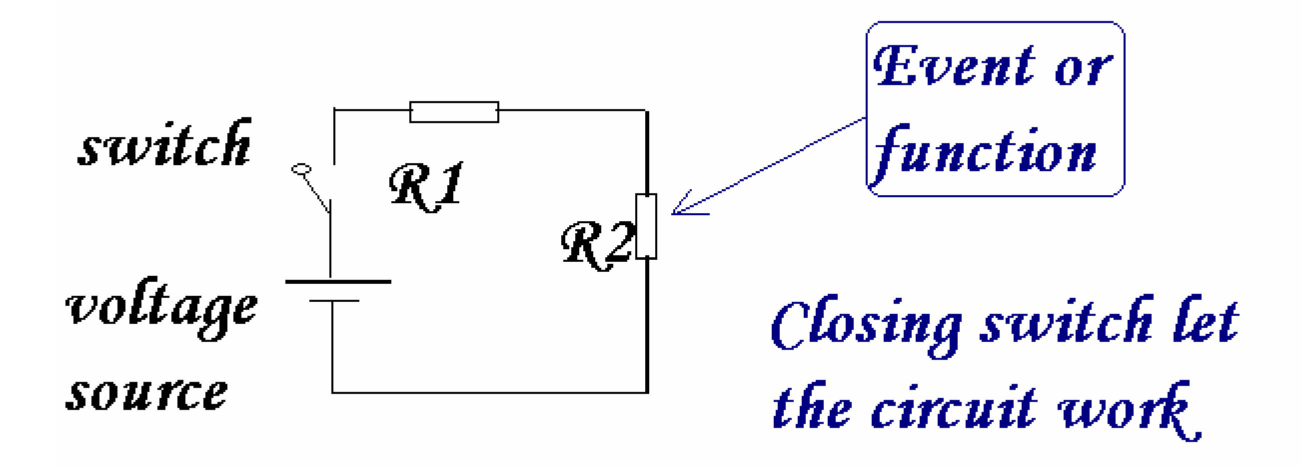
- سهل التعلم اذا كان المبرمج لديه فكره بسيطة عن الدوائر الكهربية ومفهوم الفرق بين توصيلات التوالى
  والتوازى حيث يكون التيار هنا هو ترتيب سير العمليات.
- يسهل عملية تصميم نظام مبنى على الاحداث وتوفير رسم تمثيلى للنظام من خلال عمل Circuit Diagram



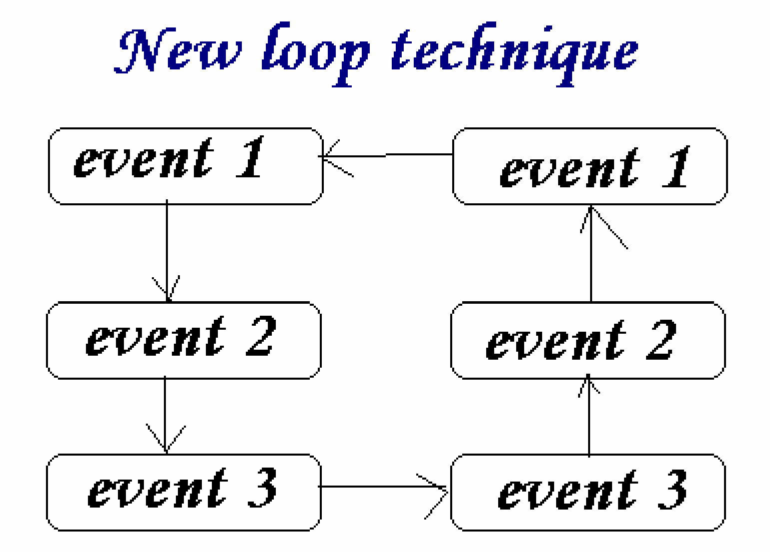
شكل (١٣) . دائرة تيار الاحداث

شكل (١٣) يبين لنا المكونات الاساسية – حيث نتخيل ان الاحداث سوف تكون فى دائرة كهربية وهذه الاحداث هى عبارة عن مقاومات فى الدائرة ودوال بالنسبة للمعالج

دائرة تيار الاحداث يمكن ان تشمل اكثر من فرع بينما يمكن ان يشمل الفرع اكثر من مقاومة ويحتوى كل فرع على مفتاح.

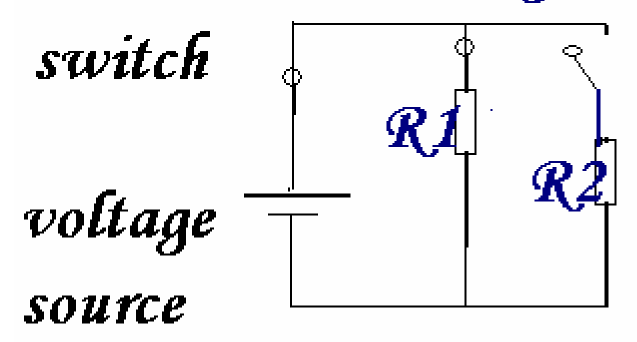


شكل (١٤) .مفهوم المقاومة (الدالة) والمفتاح



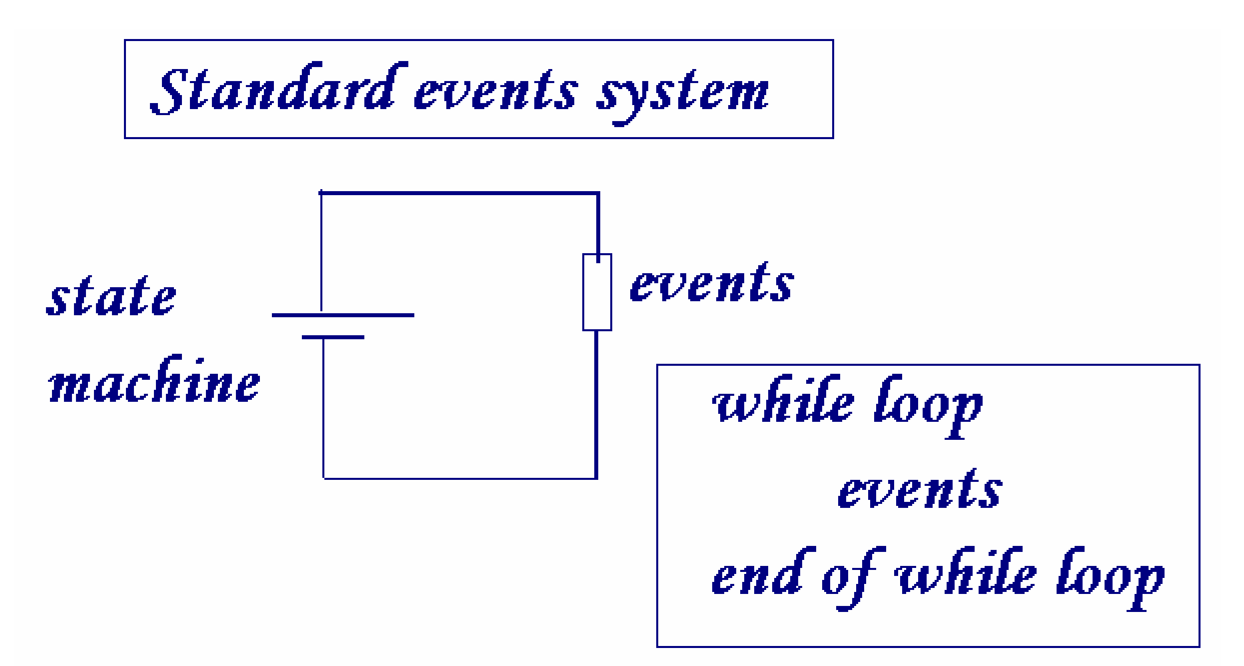
شكل (١٥) يمكن ترتيب تنفيذ الاحداث من قبل النظام بصور مختلفة

Timer event using events current circuit



R1:Timer control function
R2:Timer event function

شكل (١٦) .يمكن عمل مؤقت (تايمر) من خلال مكونات دائرة الاحداث



شكل (١٧) . الحلقة الاساسية فى نظام دائرة الاحداث

Events current circuit design components

circuit database management system

circuit events management system

شكل (١٨) لبرمجة نظام ادراة الاحداث نحتاج الى جزئين فى النظام

عند برمجة نظام ادارة الاحداث يمكن تقسيمه الى 2 Classes اى فصيلتين واحدة لتداول بيانات الدوائر والاخرى لكى تنفذ النظام وتبدا العمل على البيانات (الدوائر والفروع والمفاتيح + الدوال)

(Branches data stored in array or database file)

Branch Id	Parallel to element	Switch status
The number of the branch to identify on it	1 - branch 2 - resistance	The status of branch switch (closed is default)

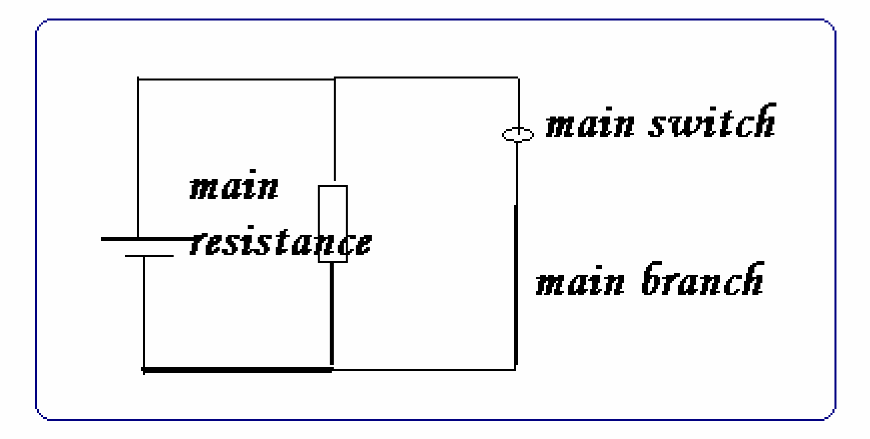
شكل (١٩)البيانات الازم تخزينها عن الفروع

Resistances data stored in array or database file

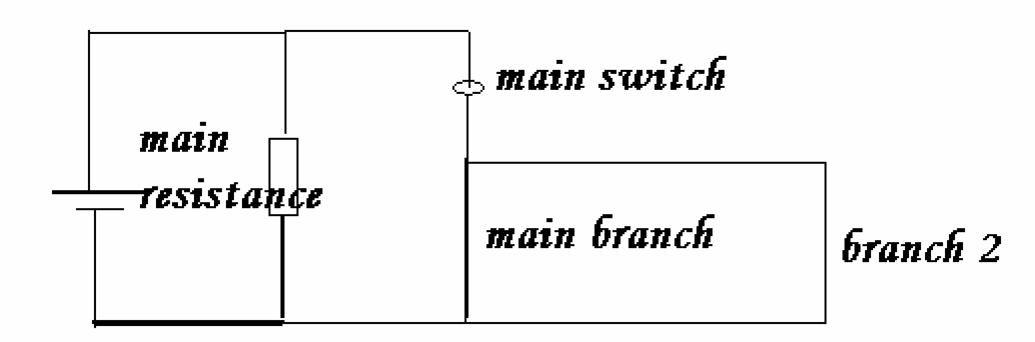
Resistance Id	Branch Id	Resistance job
The number of the resistance to identify on it	The branch which the resistance lies in it	event function name of the resistance to call

شكل (٢٠) البيانات الازم تخزينها عن الفروع

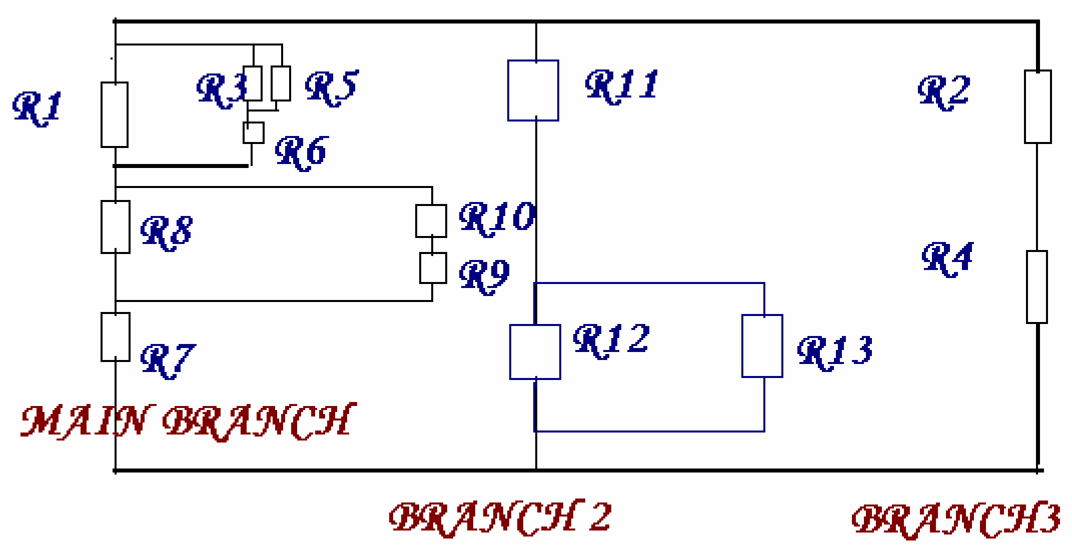
Diagram represent the status of the event current circuit at the starting of work



شكل (٢١) مكونات دائرة تيار الاحداث عند بداية انشائها



شكل (٢٢) رسم يوضح كيفية امتداد دائرة تيار الاحداث



شكل (٢٣) رسم يوضح كيفية ازدحام النظام بالمقاومات (الدوال)

وفيما يلى عرض للاكواد المستخدمة لعمل النظام – وقد تم كتابته باستحدام لغة البرمجة كليبر CA-Clipper وتم استخدام المكتبة CLASS(Y) للحصول على دعم برمجة الكائنات كما تم كتابة برنامج لاختبار النظام وشكل (٢٤) يوضح نتيجة تنفيذ هذا البرنامج حيث يتم عرض مجموعة من الرسائل بالتتابع وفى نفس الوقت يتم تشغيل تايمر لعرض الزمن الحالى بصفة مستمرة كما ان حالة النظام معلومة وتحت المراقبة (رقم الفرع – رقم المقاومة) وهكذا.

```
*---------------------------------------------------*
* CIRCUIT DATABASE CLASS
* BY,MAHMOUD SAMIR FAYED (2005)
*---------------------------------------------------*
* CALLING HEADERS FILES
* CLASS(Y).CH IS THE HEADER FILE FOR CLASSY LIBRARY
#INCLUDE "CLASS(Y).CH"
* DEFINING CLASS METHODS
CREATE CLASS GUI_CIRCUITDATA
    VAR BRANCHES
    VAR RESISTANCES
    VAR MAINSWITCH
    VAR MAINJOB
EXPORT :
    METHOD INIT
    METHOD GUI_NEWBRANCH
    METHOD GUI_NEWRESISTANCE
    METHOD GUI_SETPARALLEL
    METHOD GUI_GETPARALLEL
    METHOD GUI_SETJOB
    METHOD GUI_GETJOB
    METHOD GUI_SETBRANCH
    METHOD GUI_GETBRANCH
    METHOD GUI_SETSWITCH
    METHOD GUI_GETSWITCH
    METHOD GUI_REVERSESWITCH
    METHOD GUI_SETMSWITCH
    METHOD GUI_GETMSWITCH
    METHOD GUI_SETMJOB
    METHOD GUI_GETMJOB
END CLASS
* END OF DEFINING CLASS METHODS

METHOD INIT()
::BRANCHES      = {}
::RESISTANCES   = {}
```

```
::MAINJOB        = ""
::MAINSWITCH     = .T.
RETURN SELF

METHOD GUI_NEWBRANCH(PARA1)
LOCAL ID
ID = LEN(::BRANCHES) + 1
AADD(::BRANCHES,{ID,PARA1,.T.})
RETURN ID

METHOD GUI_NEWRESISTANCE(PARA1,PARA2)
LOCAL ID
ID = LEN(::RESISTANCES) + 1 +  1000000
AADD(::RESISTANCES,{ID,PARA1,PARA2})
RETURN ID

METHOD GUI_SETPARALLEL(PARA1,PARA2)
::BRANCHES[PARA1][2] = PARA2
RETURN SELF

METHOD GUI_GETPARALLEL(PARA1)
RETURN ::BRANCHES[PARA1][2]

METHOD GUI_SETJOB(PARA1,PARA2)
PARA1 = PARA1 - 1000000
::RESISTANCES[PARA1][3] = PARA2
RETURN SELF

METHOD GUI_GETJOB(PARA1)
PARA1 = PARA1 – 1000000
RETURN ::RESISTANCES[PARA1][3]

METHOD GUI_SETBRANCH(PARA1,PARA2)
PARA1 = PARA1 - 1000000
::RESISTANCES[PARA1][2] = PARA2
RETURN SELF

METHOD GUI_GETBRANCH(PARA1)
PARA1 = PARA1 - 1000000
RETURN ::RESISTANCES[PARA1][2]

METHOD GUI_SETSWITCH(PARA1,PARA2)
::BRANCHES[PARA1][3] = PARA2
RETURN SELF
```

```
METHOD GUI_GETSWITCH(PARA1)
RETURN ::BRANCHES[PARA1][3]

METHOD GUI_REVERSESWITCH(PARA1)
    IF ::BRANCHES[PARA1][3] = .T.
        ::BRANCHES[PARA1][3] = .F.
    ELSE
        ::BRANCHES[PARA1][3] = .T.
    ENDIF
RETURN SELF

METHOD GUI_SETMSWITCH(PARA1)
    ::MAINSWITCH = PARA1
RETURN SELF

METHOD GUI_GETMSWITCH()
RETURN ::MAINSWITCH


METHOD GUI_SETMJOB(PARA1)
    ::MAINJOB = PARA1
RETURN SELF

METHOD GUI_GETMJOB()
RETURN ::MAINJOB
*----------------------------------------------------*
* CIRCUIT EVENTS CLASS
* BY,MAHMOUD SAMIR FAYED (2005)
*----------------------------------------------------*
* CALLING HEADERS FILES
* CLASS(Y).CH IS THE HEADER FILE FOR CLASSY LIBRARY
#INCLUDE "CLASS(Y).CH"
* DEFINING CLASS METHODS
CREATE CLASS GUI_EVENTS
EXPORT :
    VAR IDLEARRAY
    VAR DIRECTION
    VAR SHUTDOWN
    VAR DATAOBJ
    METHOD INIT
    METHOD GUI_FIREON
    METHOD GUI_FIREOFF
    METHOD GUI_ADDEVENT
    METHOD GUI_DOEVENTS
```

```
     METHOD GUI_DOBRANCH
     METHOD GUI_DORESISTANCE
     METHOD GUI_SETDIRECTION
     METHOD GUI_GETIDTYPR
     METHOD GUI_GETORDER
         END CLASS
* END OF DEFINING CLASS METHODS

METHOD INIT(PARA1)
::IDLEARRAY = {}
::DIRECTION = 1
::SHUTDOWN  = .F.
::DATAOBJ   = PARA1
RETURN SELF

METHOD GUI_FIREON()
LOCAL WAY,R,MYRESISTANCES,X,BID,BLIST,T
PRIVATE PS1,PS2,PS3
WAY = ::DIRECTION
DO WHILE ::SHUTDOWN = .F.
R = ::DATAOBJ:GUI_GETMJOB()
R = &R
        IF ::SHUTDOWN = .T.
                EXIT
      ELSE
         IF ::DATAOBJ:GUI_GETMSWITCH() = .T.
              MYRESISTANCES = ::GUI_GETORDER()
              IF WAY = 1
                  FOR  X = 1 TO LEN(MYRESISTANCES )

                    PS1 = MYRESISTANCES[X][2]
                    PS2 = MYRESISTANCES[X][1]
                    PS3 = 1

                    R = ::DATAOBJ:GUI_GETMJOB()
                    R = &R
                    IF ::SHUTDOWN = .T.
                            EXIT
                    ENDIF

                 * RUNING RESISTANCE IF SWITCH CLOSED
                    BID   =  MYRESISTANCES[X][2]
                    BLIST = ::DATAOBJ:BRANCHES
                    FOR T = 1 TO LEN(BLIST)
                        IF BLIST[T][1] = BID
```

```
                IF BLIST[T][3] = .T.
                   R =  MYRESISTANCES[X][3]
                  R = &R
                   EXIT
                ENDIF
               ENDIF
           NEXT

*****************************************

         * PARAMETERS TO MAIN RESISTANCE
           PS1 = MYRESISTANCES[X][2]
           PS2 = MYRESISTANCES[X][1]
           PS3 = 2

*****************************************

           R = ::DATAOBJ:GUI_GETMJOB()
           R = &R
           IF ::SHUTDOWN = .T.
                   EXIT
           ENDIF
         NEXT

     ELSE

  FOR  X =  LEN(MYRESISTANCES ) TO 1 STEP -1

           PS1 = MYRESISTANCES[X][2]
           PS2 = MYRESISTANCES[X][1]
           PS3 = 1

           R = ::DATAOBJ:GUI_GETMJOB()
           R = &R
           IF ::SHUTDOWN = .T.
                   EXIT
           ENDIF

         * RUNING RESISTANCE IF SWITCH CLOSED
           BID   =  MYRESISTANCES[X][2]
           BLIST = ::DATAOBJ:BRANCHES
           FOR T = 1 TO LEN(BLIST)
               IF BLIST[T][1] = BID
                IF BLIST[T][3] = .T.
                   R =  MYRESISTANCES[X][3]
```

```
                        R = &R
                        EXIT
                     ENDIF
                    ENDIF
                NEXT

*****************************************

             * PARAMETERS TO MAIN RESISTANCE
                PS1 = MYRESISTANCES[X][2]
                PS2 = MYRESISTANCES[X][1]
                PS3 = 2

*****************************************

                R = ::DATAOBJ:GUI_GETMJOB()
                R = &R
                IF ::SHUTDOWN = .T.
                        EXIT
                ENDIF

              NEXT

          ENDIF

          IF .NOT. WAY = ::DIRECTION
                IF .NOT. ::DIRECTION = 3
                        WAY = ::DIRECTION

                ELSE
                        IF WAY = 1
                                WAY = 2
                        ELSE
                                WAY = 1
                        ENDIF
                ENDIF
          ENDIF
        ENDIF
    ::GUI_DOEVENTS()
    ENDIF
ENDDO
RETURN SELF
```

```
METHOD GUI_FIREOFF()
::SHUTDOWN = .T.
RETURN SELF

METHOD GUI_ADDEVENT(PARA1)
AADD(::IDLEARRAY,PARA1)
RETURN SELF

METHOD GUI_DOEVENTS()
LOCAL X , R
FOR X = 1 TO LEN(::IDLEARRAY)
R = ::IDLEARRAY[X]
R = &R
NEXT
RETURN SELF

METHOD GUI_DOBRANCH(PARA1)
LOCAL MYARR,X,R
MYARR = ::DATAOBJ:RESISTANCES
FOR X = 1 TO LEN(MYARR)
 IF MYARR[X][2] = PARA1
   R = MYARR[X][3]
   R = &R
 ENDIF
NEXT
RETURN SELF

METHOD GUI_DORESISTANCE()
LOCAL MYARR,X,R
MYARR = ::DATAOBJ:RESISTANCES
FOR X = 1 TO LEN(MYARR)
 IF MYARR[X][1] = PARA1
   R = MYARR[X][3]
   R = &R
   EXIT
 ENDIF
NEXT
RETURN SELF

METHOD GUI_SETDIRECTION(PARA1)
::DIRECTION = PARA1
RETURN SELF

METHOD GUI_GETIDTYPE(PARA1)
```

```
IF PARA1 < 1000000
RETURN 1
ELSE
RETURN 2
ENDIF
RETURN SELF

METHOD GUI_GETORDER()
* IN THIS FUNCTION WE DON'N USE THE ALGORITHM
* BECAUSE WE DON'N NEED THE ORDER IN RUNNING
*RESISTANCES IN OUR GUI SYSTEM
RETURN ::dataobj:resistances


*Test Program , Mahmoud Fayed
#INCLUDE "CLASS(Y).CH"
DO CIRCUIT
DO CIRCUIT2
SET CURSOR OFF
SET SCOREBOARD OFF
SET COLOR TO BG+/B
CLEAR
MYCIRCUIT := GUI_CIRCUIT():NEW()
MYCONTROL := GUI_EVENTS():NEW(MYCIRCUIT)
MYCONTROL:DIRECTION = 1
B1 = MYCIRCUIT:GUI_NEWBRANCH(0)
B2 = MYCIRCUIT:GUI_NEWBRANCH(1)
R1 = MYCIRCUIT:GUI_NEWRESISTANCE(B1,"HELLO()")
R2 = MYCIRCUIT:GUI_NEWRESISTANCE(B1,"HELLO2()")
R3 = MYCIRCUIT:GUI_NEWRESISTANCE(B2,"HELLO3()")
R4 = MYCIRCUIT:GUI_NEWRESISTANCE(B2,"HELLO4()")
R5 = MYCIRCUIT:GUI_NEWRESISTANCE(B2,"HELLO5()")
R6 = MYCIRCUIT:GUI_NEWRESISTANCE(B2,"HELLO6()")

MYCIRCUIT:GUI_SETMJOB("CONTROL()")
MYCONTROL:GUI_ADDEVENT("SHOWTIME()")
MYCONTROL:GUI_FIREON()

SET COLOR TO W/N
CLEAR
FUNCTION HELLO()
@5,10 SAY ;
"MAHMOUD SAMIR FAYED EVENTS CURRENT CIRCUIT  SYSTEM"
RETURN
```

```
FUNCTION HELLO2()
WAITTIME(.5)
RETURN

FUNCTION HELLO3()
@5,0 CLEAR TO 5,79
RETURN

FUNCTION HELLO4()
WAITTIME(.5)
RETURN

FUNCTION HELLO5()
@5,10 SAY "             YEAR 2005 , FAYEDCOM      "
RETURN

FUNCTION HELLO6()
WAITTIME(.5)
RETURN

FUNCTION SHOWTIME()
@1,1 SAY "TIME:" + TIME()
I = INKEY()
IF LASTKEY() <> 0
SET COLOR TO W/N
CLEAR
QUIT
ENDIF
RETURN

FUNCTION CONTROL()
@20,1 SAY "BRANCH     ID                : "
?? PS1
@21,1 SAY "RESISTANCE ID                : "
?? PS2
@22,1 SAY "STATUS(1 - BEFORE, 2 AFRTER) : "
?? PS3
WAITTIME(.5)

FUNCTION WAITTIME(PARA1)
N = SECONDS()
DO WHILE SECONDS() - N < PARA1
MYCONTROL:GUI_DOEVENTS()
ENDDO
```

```
TIME:01:44:34

YEAR 2005 , FAYEDCOM

CIRCUIT CONTROL TEST PROGRAM

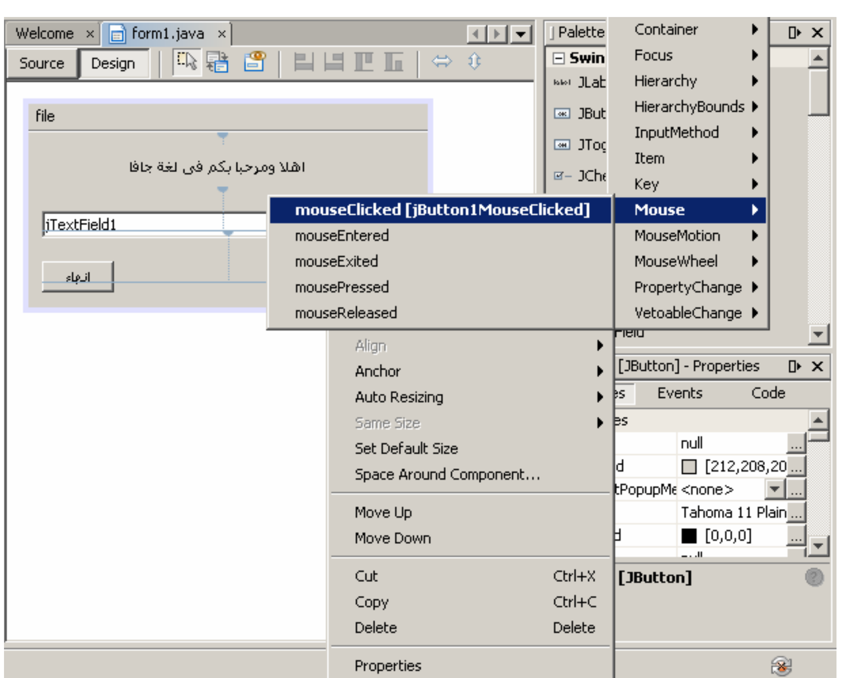
BRANCH      ID                  :        2
RESISTANCE  ID                  :  1000006
STATUS(1 - BEFORE, 2 AFRTER)    :        1
```

شكل (٢٤) برنامج اختبار نظام ادارة الاحداث المبنى على الدوائر الكهربية

ملحوظة هامة

يجب دائما ان نتذكر اننا لا نحتاج لتصميم نظام ادارة الاحداث من الصفر عند تطوير التطبيقات المتطورة بل نستخدم النظام المتوفر من قبل لغة البرمجة ويعمل بمساندة نظام التشغيل – ويكون ذلك متاح من خلال IDE الخاصة بلغة البرمجة المتطورة انظر الشكل التالى والذى يوضح ذلك من خلال NetBeans IDE 5.5 الخاصة بلغة البرمجة JAVA

شكل(٢٥) : محيط تطوير تطبيقات لغة جافا



شكل(٢٦) : اختيار حدث معين لكتابة الكود به

```
Welcome ×   form1.java ×
Source   Design

    private void jMenuItem1MouseClicked(java.awt.event.MouseEvent evt) {

    }

    private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
        System.exit(0) ;
    }

    private void jButton1MouseClicked(java.awt.event.MouseEvent evt) {
// TODO add your handling code here:
      jLabel1.setText("hello "+  jTextField1.getText()) ;
    }

    private void btn_end_click(java.awt.event.MouseEvent evt) {
// TODO add your handling code here:
      System.exit(0);
    }

    /**
     * @param args the command line arguments
```

شكل(٢٧) : كتابة التعليمات

الباب الثانى
نمط البرمجة

مقدمة هامة:-

مرحبا بك عزيزى القارىء فى اهم عنصر من عناصر تصميم النظام – الا وهو نمط البرمجة والذى يقصد به الاسلوب العام المتبع فى تصميم مكونات النظام المختلفة لكى تعمل معا بصورة متجانسة لتودى الهدف منها.

من خلال فهمك للباب السابق "نموذج سير العمليات" علمت ان التعليمات يتم تنفيذها واحدا تلو الاخر اذا لم يحدث ما يغير فى ذلك – فى الواقع اذا كتبت نظام عملاق بهذه الطريقة (تعليمة تلو الاخرى- بدون تقسيم النظام الى اجزاء منفصلة) فانك تكون قد اتبعت الاسلوب البدائى فى البرمجة وهو مايسمى Water Fall Method حيث تاتى تعليمات البرنامج واحدة تلو الاخرى كما يسقط الماء – ونسمى ذلك ايضا باسم Imperative Paradigm والذى يرتكز على Do This then do That بمعنى نفذ التعليمة الحالية التى تقف عندها ثم نفذ التعليمة التى تليها – فى الواقع لاجديد هنا فذلك المفهوم البدائى لعملية البرمجة

لم يكتفى علماء الحاسب بهذا الفكر البدائى فى كتابة البرامج – بل امتد نظرهم الى المبانى العملاقة ملتمسين الكثير من فكر المهندسين المعماريين – ان المبانى العملاقة مقسمة الى ادوار وغرف واعمدة وهكذا وهذا يسهل الاشارة الى اى مكان فى المبنى – وظهر السوال

س :- لماذا لاتكون البرامج مثل المبانى فى المفهوم ؟

جـ :- لما لا ولكن س:- ما الفائدة ؟

جـ :- ببساطة يسهل الاشارة الى اى جزء فى البرنامج العملاق – كما انه يكون اكثر تنظيما وسهل الفهم والمتابعة

ومن هنا نشات فكرة البرمجة الهيكلية Structure Programming او مايطلق عليه Functional Paradigm

ولم يصل الامر لهذا الحد بل امتد نظر العلماء الى طريقة التعامل بين البشر (الكائنات) من حيث تبادل الرسائل والمعلومات والخدمات مع مراعاة السرية والخصوصية وغيرها- نشات فكرة لما لا تقسم اجزاء البرنامج بهذه الطريقة كصورة مبدئية والتى تطورت لتشمل مفاهيم اكثر فعالية مثل الوراثة وغيرها. وبهذا نشات فكرة برمجة الكائنات التى لاقت انتشارا كبيرا وادت الى تصنيع برمجيات ذات مكونات يمكن اعادة استخدامها بسهولة.

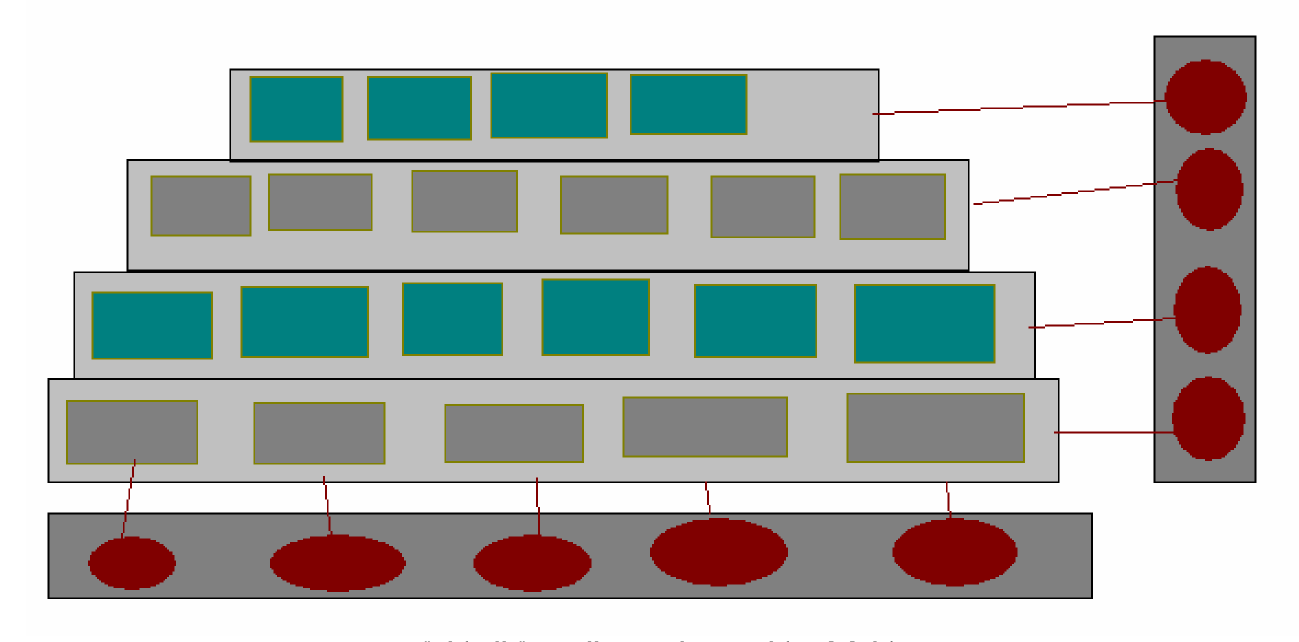
ولكن الامر لم يقف عند هذا الحد فمازال العقل البشرى فى عمل مستمر باحثا عن كل جديد مفيد ومن هنا اتجه العلماء الى ابتكار انماط برمجة جديدة لكى تحل مشاكل العصر والتى لا ياخذها نمط برمجة الكائنات فى الاعتبار ومن الانماط الجديدة

التى لم تستقر فى تكنولوجيا البرمجيات بعد – نمط برمجة العميل الموجه Agent Oriented Programming ونمط برمجة اللغات الموجهة Language Programming Oriented ونمط برمجة الخادم الممتاز (Super Server) DoubleS وهو من ابتكار مولف هذا الكتاب.

ان نمط البرمجة يتعلق بالكيف (اى كيف يتم صناعة البرمجيات) ولا يتعلق بماذا (ماذا يتم تصعنيه) فمن المفترض ان اى نمط برمجة يمكنه صناعة اى نوع من البرمجيات – ولكن يكمن الاختلاف فى الاسلوب الذى يوثر بصورة كبيرة على الزمن الازم للتطوير – قابلية النظام للفهم – قابلية النظام للتطوير

ان اتقان نمط البرمجة المتاح امر ضرورى للتواجد بين المبرمجين – واحتراف نمط البرمجة امر هام جدا عند تطوير النظم المعقدة واختيار نمط البرمجة المناسب امر حيوى جدا ولكن هذا الكتاب لايقف عند تلك النقطة الخاصة بفهم نمط البرمجة بل يمتد ليشمل كيفية صناعة نمط البرمجة مما يفتح افق القارى نحو الابتكار ويثبت اقدامه على عرش الاحتراف.

البرمجة الهيكلية



شكل(١) شكل مبسط يوضح البرمجة الهيكلية

بالنظر الى شكل (١) وهو شكل مقترح لتوضيح المفهوم تخيل ان النظام مقسم الى مجموعة من المستويات بحيث ان كل مستوى مقسم الى مجموعة من الاجزاء المجتمعة معا وقد تكون مرتبطة ببعضها (تعتمد على بعضها البعض) او مستقلة.

ولكن يتضح من الرسم ان النظام يشمل مجموعة من المستويات والتى تجتمع معا فى نظام واحد وقد تكون مرتبطة ببعضها او مستقلة عن بعضها البعض

ويوجـد علـى اليمـين مسـارات توضـح انـه يمكـن الاتصـال الراسـى بـين اجـزاء النظـام (المستويات المختلفة) ويوجد فى الاسفل مسارات توضح انـه يمكـن الاتصـال الافقـى بين اجزاء النظام (مكونات المستوى الواحد).

المراد من ذلك انه فى البرمجة الهيكلية يتم تقسيم النظام اى مكونات صغيرة (الـدوال Functions) ويتم تجميع كل مجموعة من الدوال معا فى مسـتوى واحـد (Procedure or Program File) وهذه الدوال المجتمعة معا لا يشترط ان تعتمد على بعضها البعض ولكن يفترض ان يجمعها مفهوم او معنى متضامن لذلك تم تجميعها معا.

ويمكن ان يشتمل النظام على عدة مستويات بمعنـى مجموعـة مـن ملفـات الاجـراءات التى تحتوى دوال وقد تكون ملفات الاجراءات مرتبطـة معـا او قـد تكـون منفصـلة عـن بعضها.

الاجراء Procedure هو وحدة البرمجة الهيكليـة Structure Programming والدالـة هـى وحدة Functional Programming والفرق بـين الاجـراء والدالـة فـرق اكـاديمى وهـو ان الدالة هنا تشبه الدالة فى الرياضيات فهى ترجع قيمة عند مناداتها ولكن الاجراء لا يرجع قيمة عند مناداته

حيث ان Functional Paradigm هى ضمنيا جزء من Structure Programming فـيمكن ان تعتبر الاثنان وجهان لعملة واحدة.

ومع ان تصميم البـرامج باسـتخدام البرمجـة الهيكليـة امـر بسـيط نسـيبا الا انـه هنـاك العديد من الجوانب التى ينبغى مراعاتها عند تصميم النظام وهذه العوامل هى :-

١ – تحديد مستويات النظام
٢ – تحديد كيفية التداخل بين اجزاء النظام
٣ – تحديد انواع الدوال داخل النظام

اولا : مستويات النظام

بمعنى ربط كل مجموعة من الـدوال معـا ذات المفهـوم المتضـامن لتكـون مسـتوى مـن مستويات النظام - ومستويات النظام انواع - انظر شكل(٢)

١ – مستوى مستقل
وهو يتعاون مع بقية المستويات الاخرى ولكن لا يعتمد على اى منها.
وفى هذه الحالة من الممكن ان يكون هذا المستوى عبارة عن Sub System
وفى حالة كون المستوى المستقل يمكن استخدامه فى نظم اخرى بدون التاثير على مفهوم المعنى المتضامن فانه يطلق عليه Embedded System

٢ - مستوى غير مستقل
وهو يعتمد على مستويات اخرى داخل النظام وعندها يسمى Layer of System

ثانيا : تحديد كيفية التداخل بين اجزاء النظام

من المفترض انه بعد تقسيم النظام الى اجزاء (ترتبط معا من خلال مفهوم متضامن) ينبغى تحديد كيف ستتعاون اجزاء النظام معا واذا افترضنا ان كل جزء من اجزاء النظام عبارة عن Procedure يشمل محموعة من الدوال Functions فانه يجب تحديد المعلومات التالية لكل جزء

١ – ما هى الاجزاء التى سوف يتعاون معها هذا الجزء من النظام
٢ - ما هى المتطلبات التى سوف يحتاجها هذا الجزء من بقية اجزاء النظام
٣ - ما هى الخدمات التى سوف يقدمها هذا الجزء لبقية اجزاء النظام
٤ - تحديد نوعية الاتصال بين اجزاء النظام المختلفة – هل هى من جهة واحدة Simplex او من جهتين Duplex

ثالثا : تحديد انواع الدوال داخل النظام

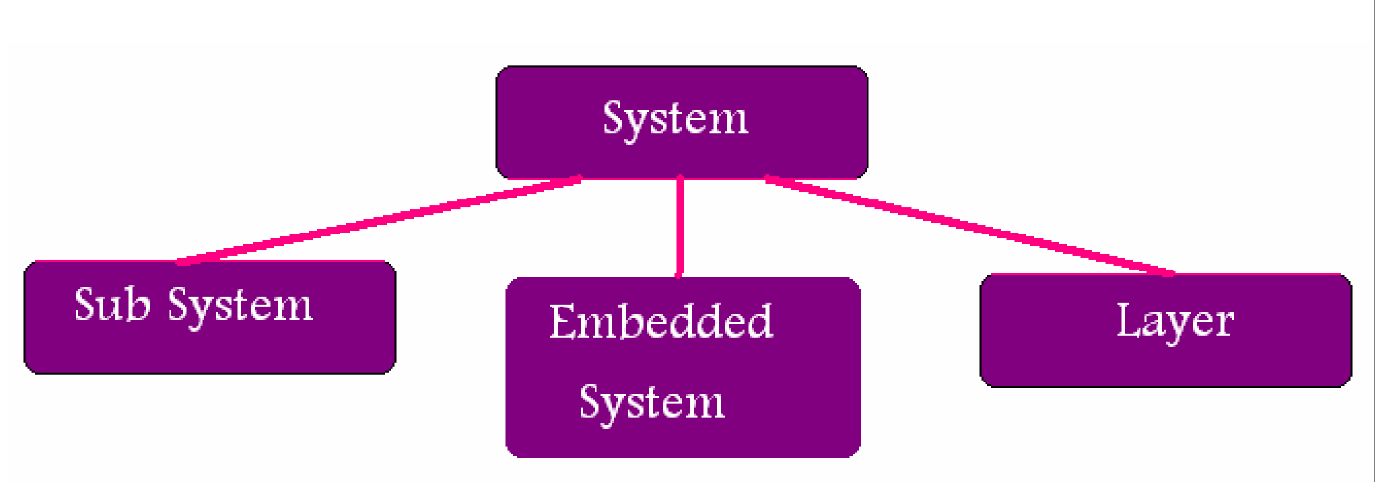
من وجهة نظر الدالة :-

- ١ – اذا كانت الدالة لا تعتمد على دوال اخرى او متغير معرف خارج نطاق الدالة فى النظام فهى دالة مستقلة يمكن ازالتها من النظام واستخدامها فى نظام اخر مباشرة (دالة مستقلة)
- ٢ – اذا كانت الدالة تعتمد على دوال اخرى فى النظام او متغير معرف خارج نطاق الدالة فهى دالة تابعة لا يتم فصلها بمفردها من النظام واستخدامها فى نظام اخر (دالة تابعة)
- ٣ - اذا كانت الدالة يتم استدعائها من قبل دالة اخرى فانها تسمى (دالة خادمة)
- ٤ - اذا كانت الدالة لا يتم استدعائها من قبل دالة اخرى فهى تسمى(دالة خاملة)

من وجهة نظر النظام :-
- ١ – الدالة الخادمة اذا تم فصلها من النظام فسوف تحصل علىرسالة خطا Error Message
- ٢ – الدالة الخاملة فصلها من النظام لا تحصل على رسالة خطا No Error Message

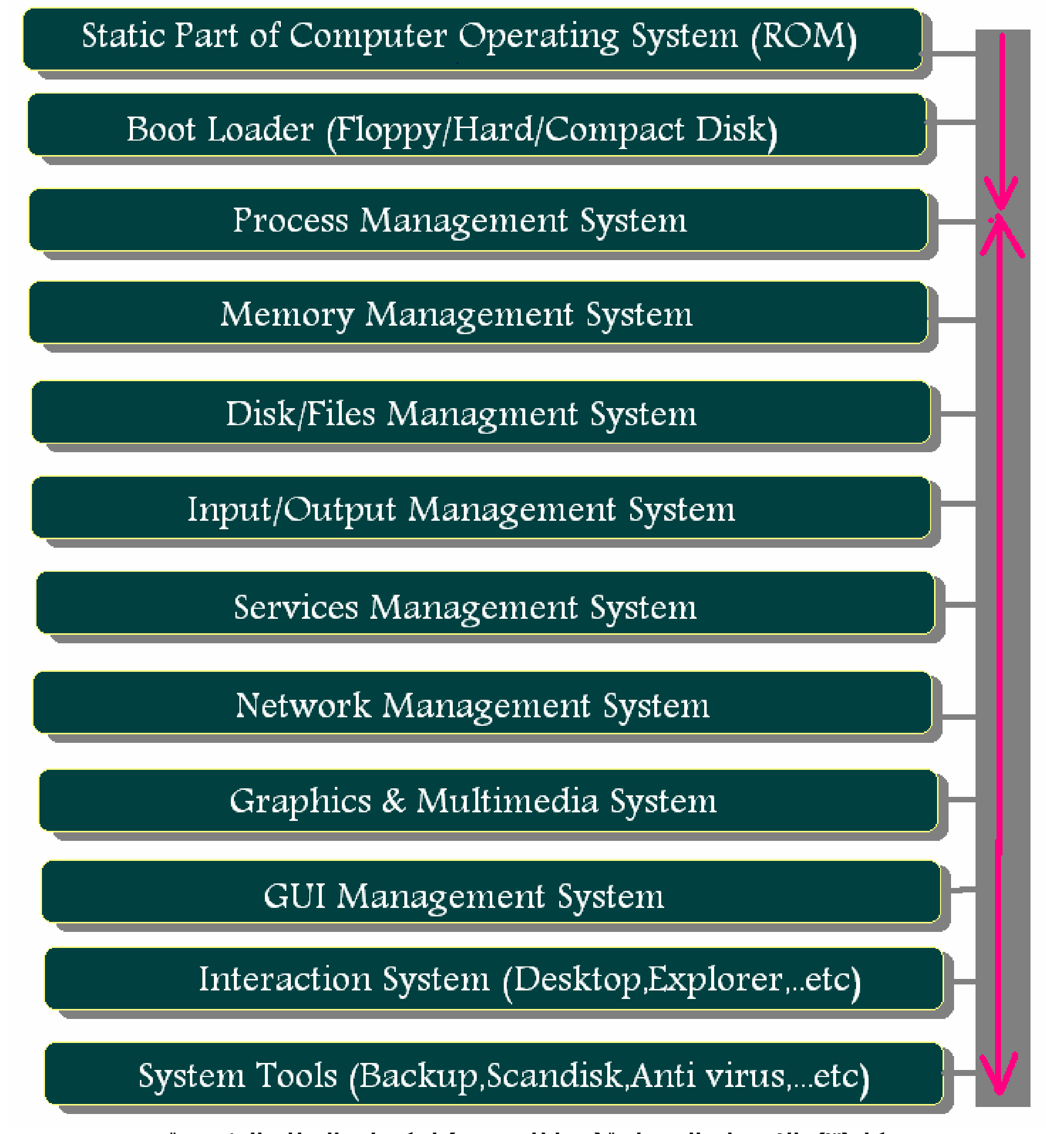
تحديد نوع النظام :-

امر فى غاية الاهمية وياتى قبل البدء فى اى شىء ونوع النظام يتعلق بمكوناته وبيئة العمل.

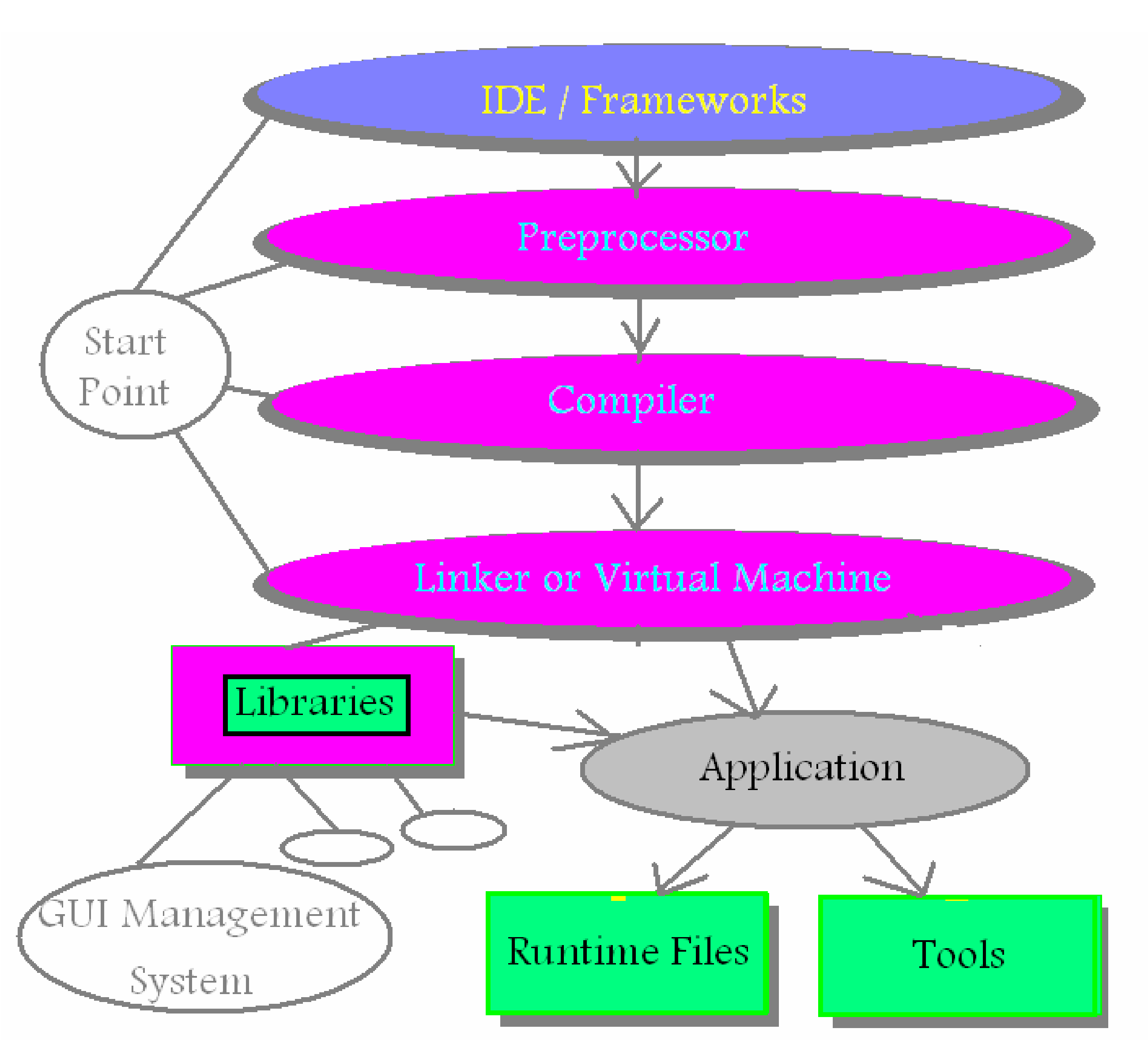


شكل(٢). الاسماء المختلفة لمكونات النظام

وسوف ناخذ الان مثال على انواع المكونات الرئسية للنظام انظر شكل (٣) الذى يوضح مكونات نظام عبارة عن نظام تشغيل كمبيوتر وانظر الى شكل (٤) والذى يوضح مكونات نظام عبارة عن لغة برمجة Programming Language



شكل(٣). الاسماء المختلفة(تبعا للمفهوم) لمكونات النظام الرئيسية



شكل(٤) : مكونات نظام عبارة عن لغة برمجة

Sub System: مثل Process Management System المتواجد ضمن مكونات نظام التشغيل ويشكل معه مفهوم متضامن

Embedded System : مثل GUI Management System والذى يتواجد ضمن مكونات نظام التشغيل ويشكل معه مفهوم متضامن الا انه يتواجد فى نظم اخرى من نوع اخر غير نظام التشغيل مثل نظام لغة البرمجة شكل (٤)

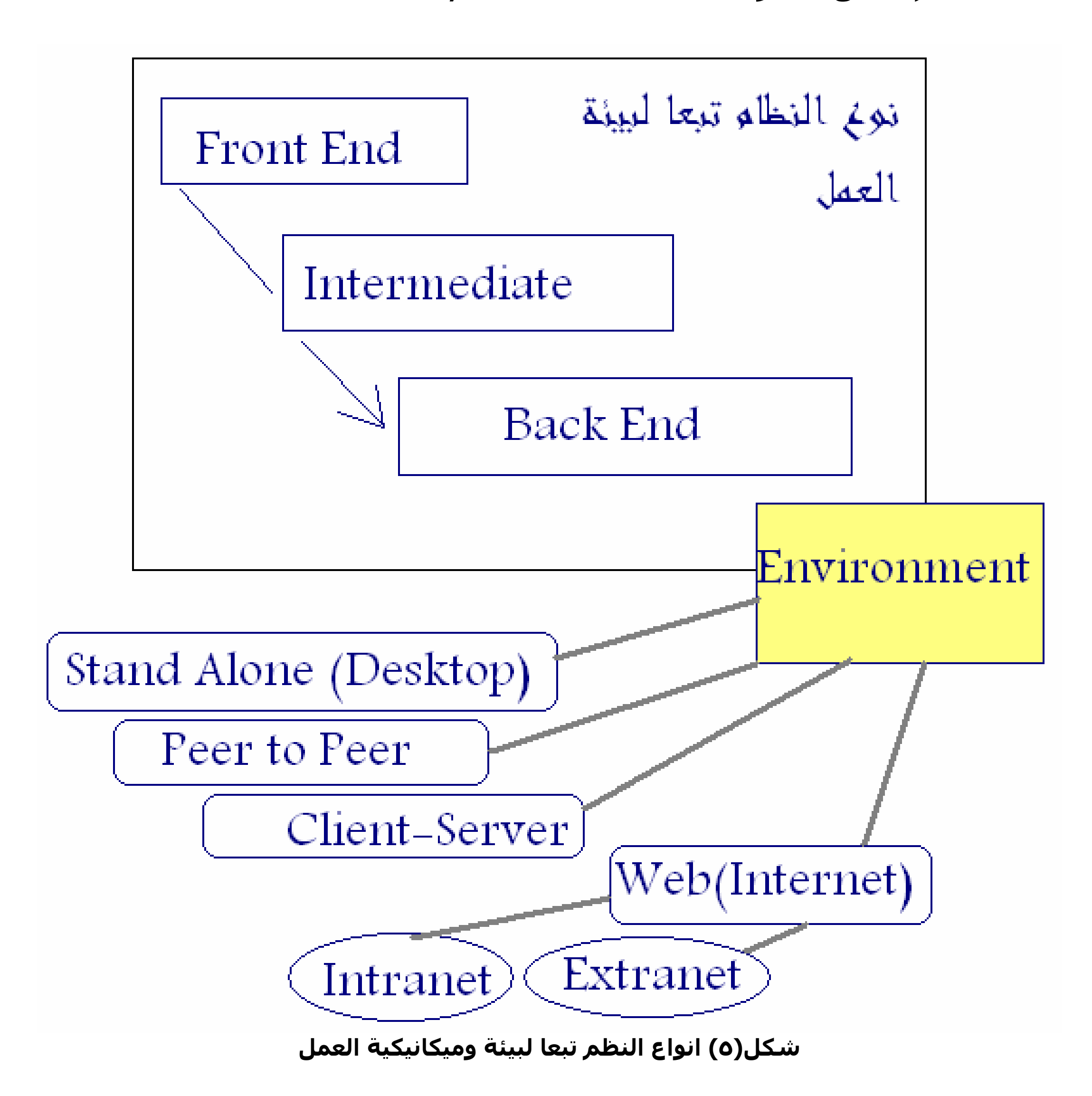
بعد تحديد هل System عبارة عن Sub System او Embedded System فان بقية الاجزاء الفرعية من النظام تكون عبارة عن Layers of System

ملحوظة هامة

ان تسمية النظام System ب Sub System او Embedded System تظهر عند النظر الى نظام كبير System والتعمق فيه بالنظر الى مكوناته

ويمكن النظر من جهة اخرى – فبدلا من النظر الى System والتعمق فيه – فاننا ننظر بعيدا من حوله ونرى البيئة التى يعمل فيها النظام System Environment وليكن النظام مثلا يعمل داخل شبكة فانه فى هذه الحالة ياخذ احد المسميات التالية Front End , Back End or Intermediate System انظر شكل (٥).

اما اذا كان النظام يعمل منفردا فانه يسمى Standalone System

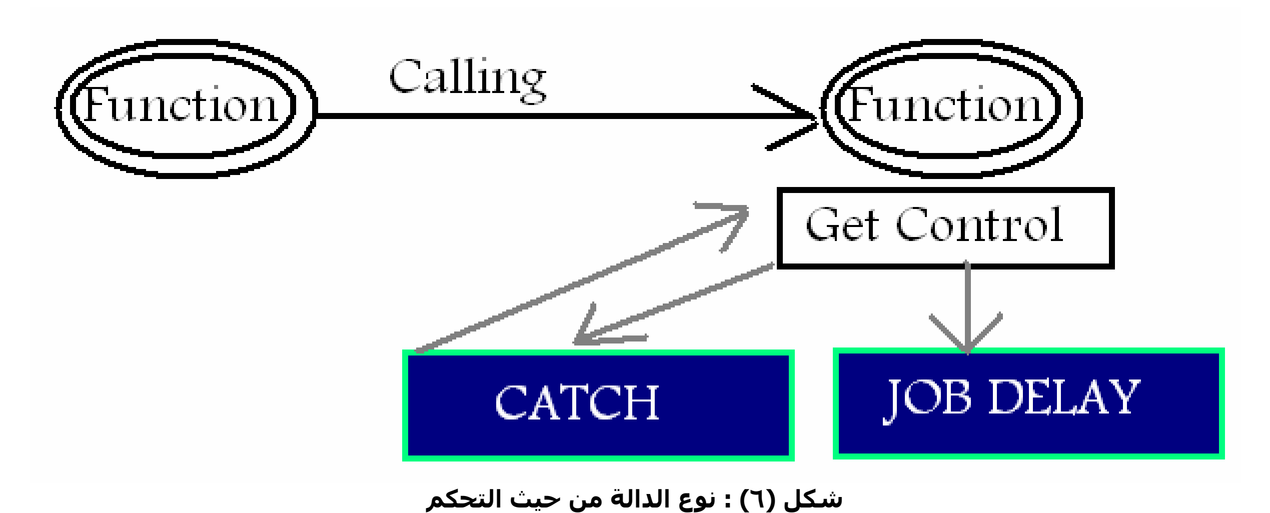


شكل(٥) انواع النظم تبعا لبيئة وميكانيكية العمل

نوع الدالة من حيث التحكم Control:-

سبق وان حددنا نوع الدالة من حيث علاقتها بالدوال داخل النظام – ولكم الان نود ان نحدد نوع الدالة من حيث التحكم بمعنى هل عند مناداة الدالة تمتلك التحكم ام لا – فى الواقع عند مناداة اى دالة فانها تحصل على التحكم Get Control ولكن ذلك لا يعنى ان الدالة تمتلك التحكم Catch Control – يقال ان الدالة تمتلك التحكم اذا كانت هذه الدالة نقطة توقف اثناء عمل النظام

ولا يقصد بذلك التوقف الزمنى Delay Time المرتبط بمهمة الدالة وانما يقصد التوقف المنطقى Logical Delay كان تحتوى الدالة على جملة While تعمل باستمرار وتعرض من خلالها قائمة خيارات فرعية مثلا – فهذا توقف منطقى غير مرتبط بالزمن فهو توقف يمكن ان يستمر الى الابد ما لم يحدث شىء يغيره – اما التوقف الزمنى فهو مرتبط بمهمة تقوم بها الدالة مثل كم هائل من العمليات الحسابية على سبيل المثال.
انظر شكل (٦) لكى يتضح المفهوم.

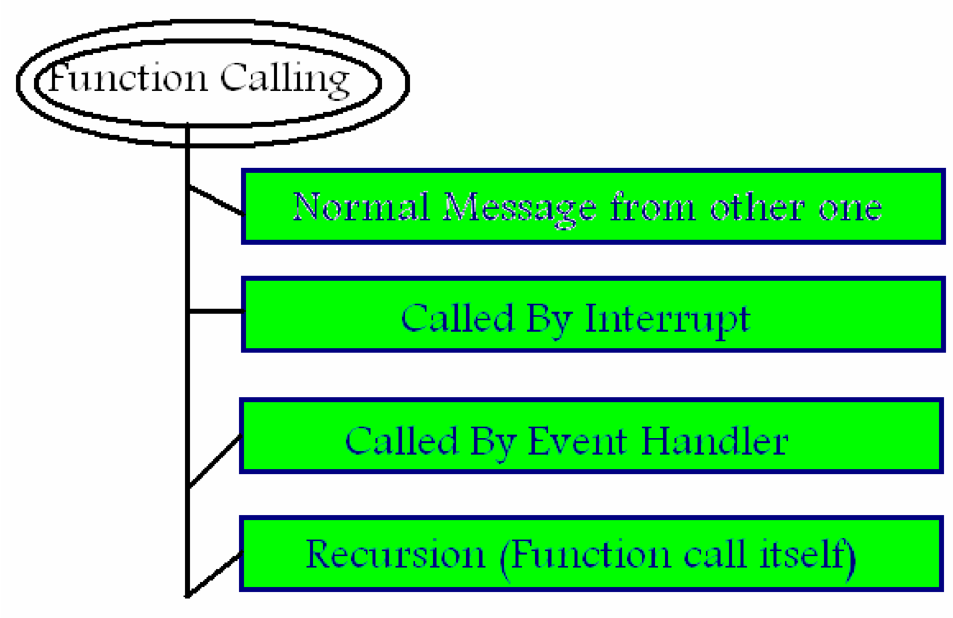


شكل (٦) : نوع الدالة من حيث التحكم

نوع الدالة من حيث مصدر التشغيل Firing Source :-

هنا يكون الاهتمام بالمصدر الذى قام بمناداة الدالة

- ١ – اما دالة اخرى
- ٢ – من خلال مقاطعة للمعالج CPU Interrupt
- ٣ - من خلال نظام ادارة الاحداث Event-Driven System
- ٤ - الدالة قامت باستدعاء نفسها

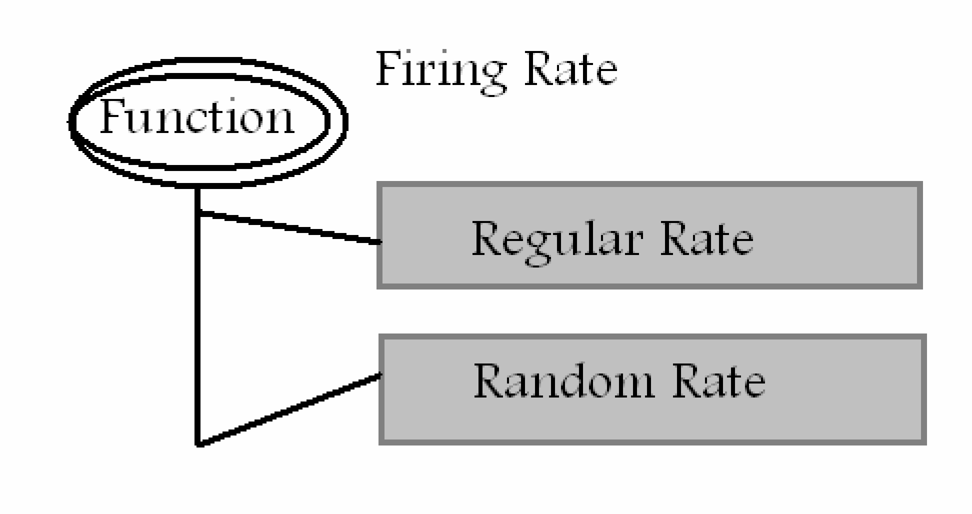


شكل(٧) : نوع الدالة من حيث التشغيل (المصدر الذى نادى الدالة)

نوع الدالة من حيث معدل التشغيل Firing Rate :-

هنا نهتم بمعدل تشغيل الدالة بمعنى هل يتم مناداة الدالة بمعدل منتظم ام لا .
ولا يقصد بالمعدل المنتظم الفترة الزمنية فقط وانما يقصد ايضا جدول المهام.
بمعنى اذا كان يتم مناداة دالة بصورة منتظمة مع اختلاف بسيط فى المعدل الزمنى وفى جدول المهام فان معدل تشغيل الدالة يقال انه منتظم مثل الدوال الموجودة داخل Queue الخاص بنظام ادارة الاحداث مثلا والمسئولة عن التحقق من حدوث الاحداث حتى تستدعى الاكواد الخاصة بها.

والعكس صحيح بالتاكيد بمعنى لو لم يكن معدل تشغيل الدالة منتظم فانه يكون غير منتظم



شكل(٨) نوع الدالة من حيث معدل التشغيل

تفاصيل استخدام الدالة :-

هنا نهتم بالمعطيات اللازمة لاستخدام الدالة (اذا وجدت) والقيمة التى ترجعها الدالة (اذا وجدت) وذلك بالتاكيد مع مهمة الدالة التى توديها

١ – مهمة الدالة
٢ – عدد المدخلات
٣ – اسم كل مدخل ونوعه وسعة التخزين Name,Type & Size
٤ – عدد المخرجات
٥ – القيمة التى ترجعها الدالة

عنوان الدالة داخل خريطة النظام:-

هنا نهتم بكيفية الوصول الى الدالة من قبل اى جزء من اجزاء النظام – فى الكثير من لغات البرمجة يكتفى بان يكون اسم الدالة هو العنوان اللازم لاستدعائها – ولكن فى التصميم ينبغى توضيع العنوان الكامل للدالة داخل خريطة النظام

حدود انتشار الدالة داخل النظام :-

بمعنى اوضح تحديد المستوى الذى تعمل فيه الدالة – ما الدوال المسموح لها بالتعامل مع الدالة وهكذا.

قابلية الدالة للتطوير :-

بمعنى فى حالة اصدار تحديثات من البرنامج Update او Patch او Service Pack – هل تخضع الدالة للتطوير ام لا – وذلك يشمل تصور خاص فى البرمجة مثل وضع الدالة فى Library اى ملف DLL ويوجد حلول اخرى مثل اعتمادية الدالة على ملفات بيانات تحمل مواصفات للغمل وهذه الملفات قابلة للتعديل وهكذا.

خطوات مهمة الدالة :-

نحدد الخطوات المنطقية لاداء مهمة الدالة باللغة العربية او الانجليزية – ثم نضع تصور عن طريق Algorithm او Flow char لكيفية برمجة الدالة.

جدول معلومات الدالة :-

هو جدول يوضح فيه تصميم الدالة ويشمل

- ١ – عنوان الدالة داخل خريطة النظام
- ٢ – نوع الدالة من حيث التحكم
- ٣ – نوع الدالة من حيث مصدر التشغيل
- ٤ - نوع الدالة من حيث معدل التشغيل
- ٥ – تفاصيل استخدام الدالة
- ٦ – الخطوات الازمة لاداء مهمة الدالة
- ٧ – حدود انتشار الدالة
- ٨ – قابلية الدالة للتطوير

مثال على البرمجة الهيكلية :-

لدينا مثال بسيط عبارة عن برنامج قاعدة بيانات صغير يعمل على جدول واحد ويقوم بعمليات الاضافة والبحث والحذف والتعديل وتم كتابته باستخدام لغة CA-Clipper (اعلم انها لغة برمجة قديمة وهذا امر طبيعى لان البرنامج عبارة عن مثال عن البرمجة الهيكلية وهى قديمة) وتم استخدام المكتبة ClipON فى البرنامج.

```
D:\WINDOWS\system32\CMD.exe - TEL

*---------------*
* Telephone 1.0 *
*---------------*

Number  :           0
Name 1  :         Name 2  :         Name 3  :         Name 4  :
Country:                    City    :
Address:                              Company:
Phone 1:            0 Phone 2:          0 Phone 3:          0
Fax 1   :           0 Fax 2   :         0 Fax 3   :         0
Note    :

   Add       Edit       Find       Delete      Report      Exit
                       add new record
```

شكل(٩) برنامج مفكرة الهاتف – تحت Dos باستخدام لغة Clipper

```
*** prog = tel.prg
*** programmer = mahmoud samir fayed
*** date = 19/5/2001
set exclusive off
c_setmode(3)     && from clipon library
*------------------------------------------------------------------*
* the program form of parts
*    part(1) = define memory variables
*    part(2) = draw screen,open tel.dbf and start main loop
*    part(3) = call m_record() function to draw record for show
*    part(4) = show main menu for user
*    part(5) = run the option that choiced by user and end main loop
*    part(6) = m_record(),m_save(),m_store() and m_cancel() function
*------------------------------------------------------------------*
*------------------------ [ part (1) ] ----------------------*
STORE 000000000000000 TO F_NUMBER
STORE 000000000000000 TO F_PHONE1
STORE 000000000000000 TO F_PHONE2
STORE 000000000000000 TO F_PHONE3
STORE 000000000000000 TO F_FAX1
STORE 000000000000000 TO F_FAX2
STORE 000000000000000 TO F_FAX3
STORE SPACE(8) TO F_NAME1
STORE SPACE(8) TO F_NAME2
```

```
STORE SPACE(8) TO F_NAME3
STORE SPACE(8) TO F_NAME4
STORE SPACE(20) TO F_COUNTRY
STORE SPACE(20) TO F_CITY
STORE SPACE(20) TO F_COMPANY
STORE SPACE(30) TO F_ADDRESS
STORE SPACE(30) TO F_NOTE
STORE "BG+/B,GR+/rB" TO M_COLOR
STORE "* Telephone 1.0 *" TO M_TEXT
STORE "*" TO M_MARK
STORE .T. TO M_SAVE
STORE "bg+/r,n/w" TO M_MSG
*---------------------- [ end of part(1) ] --------------------*
*********************************************************
*
*------------------------- [ part (2) ] -----------------------*
SET COLOR TO bg+/b,gr+/rb
@0,0 CLEAR TO 24,79
@1,32 SAY "*---------------*"
@2,32 SAY M_TEXT
@3,32 SAY "*---------------*"
@21,1 TO 23,77
@5,1 to 19,77
USE TEL shared
DO WHILE .T.
*---------------------- [ end of part(2) ] --------------------*
*********************************************************
*
*------------------------- [ part (3) ] -----------------------*
M_RECORD(.f.)
*---------------------- [ end of part(3) ] --------------------*
*********************************************************
*
*------------------------- [ part (4) ] -----------------------*
set message to 24 center
SET SCOREBOARD OFF
SET WRAP ON
SET ESCAPE OFF
@22,5  PROMPT "  Add     " message " add new record     "
@22,15 PROMPT "  Edit    " message " edit record        "
@22,25 PROMPT "  Find    " message " find a record      "
@22,35 PROMPT "  Delete  " message " delete  record     "
```

```
@22,45 PROMPT "  Report  " message " show all records   "
@22,55 PROMPT "  Exit    " message " exit of program    "
MENU TO LIST
*--------------------- [ end of part(4) ] -------------------*
**************************************************************
*
*------------------------ [ part (5) ] ----------------------*
DO CASE
  CASE LIST = 1
  SAVE SCREEN TO ADD
  M_RECORD(.T.)
  SET COLOR TO bg+/r,n/w
  @10,20 CLEAR TO 12,60
  @10,20 TO 12,60
  C_SHADOWIT(10,20,12,60,2,"w/n")
  @11,25 SAY "Save Record (T/F)" GET M_SAVE
  READ
  IF M_SAVE = .T.
  APPEND BLANK
  M_SAVE()
  m_cancel()
  ELSE
  M_CANCEL()
  ENDIF
  RESTORE SCREEN FROM ADD
  set color to bg+/b,gr+/rb
  CASE LIST = 2
  save screen to edit
  m_find()
  if .not. found()
  SET COLOR TO bg+/r,n/w
  @10,20 CLEAR TO 12,60
  @10,20 TO 12,60
  C_SHADOWIT(10,20,12,60,2,"w/n")
  @11,25 say "Record Not Found"
  inkey(15)
  else
  restore screen from edit
  set color to bg+/b,gr+/rb
  m_store()
  M_RECORD(.T.)
  SET COLOR TO bg+/r,n/w
```

```
@10,20 CLEAR TO 12,60
@10,20 TO 12,60
C_SHADOWIT(10,20,12,60,2,"w/n")
@11,25 SAY "Save Edit (T/F)" GET M_SAVE
READ
IF M_SAVE = .T.
M_SAVE()
m_cancel()
ELSE
M_CANCEL()
ENDIF
endif
restore screen from edit
SET COLOR TO BG+/B,GR+/RB
CASE LIST = 3
save screen to find
m_find()
if .not. found()
SET COLOR TO bg+/r,n/w
@10,20 CLEAR TO 12,60
@10,20 TO 12,60
C_SHADOWIT(10,20,12,60,2,"w/n")
@11,25 say "Record Not Found"
inkey(15)
else
restore screen from find
set color to bg+/b,gr+/rb
m_store()
m_record(.f.)
inkey(10)
m_cancel()
endif
restore screen from find
set color to bg+/b,gr+/rb
CASE LIST = 4
save screen to del
m_find()
if .not. found()
SET COLOR TO bg+/r,n/w
@10,20 CLEAR TO 12,60
@10,20 TO 12,60
C_SHADOWIT(10,20,12,60,2,"w/n")
```

```
  @11,25 say "Record Not Found"
  inkey(15)
  restore screen from del
  set color to bg+/b,gr+/rb
  else
  restore screen from del
  m_store()
  m_record(.f.)
  SET COLOR TO bg+/r,n/w
  @10,20 CLEAR TO 12,60
  @10,20 TO 12,60
  C_SHADOWIT(10,20,12,60,2,"w/n")
  @11,25 SAY "Delete Record (T/F)" GET M_SAVE
  READ
  IF M_SAVE = .T.
  DELETE
  PACK
  ENDIF
  restore screen from del
  set color to bg+/b,gr+/rb
  endif
  CASE LIST = 5
  save screen to report
  delete file show.txt
  report form tel plain to file show.txt
  restore screen from report
  c_view(4,6,19,73,"show.txt",2,"bg+/r",1)
  restore screen from report
  set color to bg+/b,gr+/rb
  set cursor on
  CASE LIST = 6
  set color to w/n
  clear
  cancel
END CASE

enddo
*---------------------- [ end of part(5) ] --------------------*
***************************************************************
*
*------------------------ [ part (6) ] -----------------------*
function m_record(h1)
```

```
@6,3   say  "Number :" get f_number
@8,3   say  "Name 1 :" get f_name1
@8,21  say  "Name 2 :" get f_name2
@8,41  say  "Name 3 :" get f_name3
@8,60  say  "Name 4 :" get f_name4
@10,3   say  "Country:" get f_country
@10,33  say  "City   :" get f_city
@12,3  say  "Address:" get f_address
@12,43 say  "Company:" get f_company
@14,3  say  "Phone 1:" get f_phone1
@14,23 say  "Phone 2:" get f_phone2
@14,43 say  "Phone 3:" get f_phone3
@16,3  say  "Fax 1 : " get f_fax1
@16,23 say  "Fax 2 : " get f_fax2
@16,43 say  "Fax 3 : " get f_fax3
@18,3  say  "Note  : " get f_note
if h1 = .t.
read
else
clear gets
endif
return
function m_save()
replace name1 with f_name1
replace name2 with f_name2
replace name3 with f_name3
replace name4 with f_name4
replace number with f_number
replace country with f_country
replace city with f_city
replace address with f_address
replace company with f_company
replace phone1 with f_phone1
replace phone2 with f_phone2
replace phone3 with f_phone3
replace fax1 with f_fax1
replace fax2 with f_fax2
replace fax3 with f_fax3
replace note with f_note
return
function m_store()
  STORE number TO F_NUMBER
```

```
   STORE phone1 TO F_PHONE1
   STORE phone2 TO F_PHONE2
   STORE phone3 TO F_PHONE3
   STORE fax1   TO F_FAX1
   STORE fax2   TO F_FAX2
   STORE fax3   TO F_FAX3
   STORE left(name1,8) TO F_NAME1
   STORE left(name2,8) TO F_NAME2
   STORE left(name3,8) TO F_NAME3
   STORE left(name4,8) TO F_NAME4
   STORE country TO F_COUNTRY
   STORE city TO F_CITY
   STORE company TO F_COMPANY
   STORE address TO F_ADDRESS
   STORE note TO F_NOTE
return
function m_cancel()
STORE 000000000000000 TO F_NUMBER
STORE 000000000000000 TO F_PHONE1
STORE 000000000000000 TO F_PHONE2
STORE 000000000000000 TO F_PHONE3
STORE 000000000000000 TO F_FAX1
STORE 000000000000000 TO F_FAX2
STORE 000000000000000 TO F_FAX3
STORE SPACE(8) TO F_NAME1
STORE SPACE(8) TO F_NAME2
STORE SPACE(8) TO F_NAME3
STORE SPACE(8) TO F_NAME4
STORE SPACE(20) TO F_COUNTRY
STORE SPACE(20) TO F_CITY
STORE SPACE(20) TO F_COMPANY
STORE SPACE(30) TO F_ADDRESS
STORE SPACE(30) TO F_NOTE
return
function m_find()
   SET COLOR TO bg+/r,n/w
   @10,20 CLEAR TO 12,60
   @10,20 TO 12,60
   C_SHADOWIT(10,20,12,60,2,"w/n")
   g_number = 000000000000000
   @11,25 SAY "Number :" GET g_number
   READ
```

```
locate for number = g_number
return
```

---------------------- [end of part(6)] --------------------

تمرين

"قم بعمل جدول معلومات الدوال التى يتكون منها البرنامج "

برمجة الكائنات :-

تعرضنا للبرمجة الهيكلية وكيفية وضع تصميم للنظام باستخدامها – والجدير بالذكر ان البرمجة الهيكلية تركز على JOB اى FUNCTION اثناء التصميم ولكنها لا تركز على البيانات Data - من هنا جاءت احد الافكار لماذا لا يتم ربط البيانات Data والوظيفة التى تعمل عليها معا Function ليشكلان معا مفهوم متضامن من البيانات والدوال التى تتعامل مع هذه البيانات

الفصيلة CLASS:-
عبارة عن هيكل يشمل مجموعة من البيانات و الدوال التى تعمل عليها بحيث يشكلان معا مفهوم متضامن وتسمى البيانات Attributes او Variables او Properties او Data حسب لغة البرمجة وتسمى الدوال Methods بدلا من Functions

فى البداية كان يمكن ان نقول ان CLASS ماهى الا بديل لـ Procedure فى البرمجة الهيكلية والدالة ماهى الا بديل ل Function فى البرمجة الهيكلية – هذا صحيح اذا ما توقفنا عند مفهوم الـ Class البدائى ولكن الامر لا يقف عند هذا الحد بل يمتد ليقدم لنا مفاهيم اخرى (الكبسولة – الكائن – الوراثة – التعدد)

الكبسولة Encapsulation :-
هى عملية تجميع البيانات والداول معا فى هيكل واحد – او فصيلة واحدة

الكائن Object :-
ببساطة نتخيل ان الفصيلة Class عبارة عن Data Type والكائن هو متغير من هذه ال Class ولكن الاختلاف ان هذا المتغير لا يشير فقط الا بيانات بل يشير ايضا الى دوال تعمل على هذه البيانات

الوراثة Inheritance :-
يشتمل النظام على اكثر من Class وقد نحتاج لتكرار محتويات Class مع عمل بعض التعديلات البسيطة او اضافة بعض البيانات او الدوال – فبدلا من تكرار التعليمات – نشات فكرة الوراثة وهى ان تنشى فصيلة جديدة تحمل كل صفات الفصيلة الام ثم بعد ذلك تقوم بعمل الاضافات التى تريدها

التعدد Polymorphism :-
حيث ان الفصيلة اضافت مستوى جديد لعنوان الدالة – بمعنى ان Method لا يشار اليها مباشرة وانما يتم الاشارة اليها من خلال Class او من خلال Object – من هنا اصبح يمكن ان تجد اكثر من Method بنفس الاسم فى اكثر من Class مختلفة

التركيب المتداخل Composition :-
ويقصد به ان يكون احد Attribute الذى تشتمل عليه ال Class عبارة عن كائن Object من Class اخرى

التفويض Delegation :-
ويقصد به ان تنادى Method من خلال عنوانها مـن قبـل Method اخـرى – وتظهـر هنـا تعقيدات الحصول على عنوان Method التى تقوم بمناداتها.

ملحوظة هامة

" برمجة الكائنات ادت الى ان تصميم النظم يتم من خلال تقسيم النظام الى عناصر ذات مدلول له معنى واكثر تضامنا – وفكرة برمجة الكائنات ترتكز على تخيل عناصر النظام كمجموعة من العناصر (كائنات) التى لها سمات مشتركة (فصائل) وتتبادل الطلبات Messages فيما بينها "

فنيات برمجة الكائنات :-

١ – تقسيم العناصر المتكررة فى النظام الى فصائل رئيسية واستخدام الوراثة لاعادة استخدامها فى صورة اخرى غير الصورة الرئيسية التى نشات عليها يوفر الكثير من الوقت

٢ - استخدام التركيب المتداخل والتفويض يعنى مرونة هائلة منقطعة النظير وهى فائدة عظيمة لا يعرفها الا المحترفين من مبرمجين النظم والجدير بالذكر ان التركيب المتداخل والتفويض من الممكن ان يستخدمان كبديل للوراثة.

ملحوظة هامة

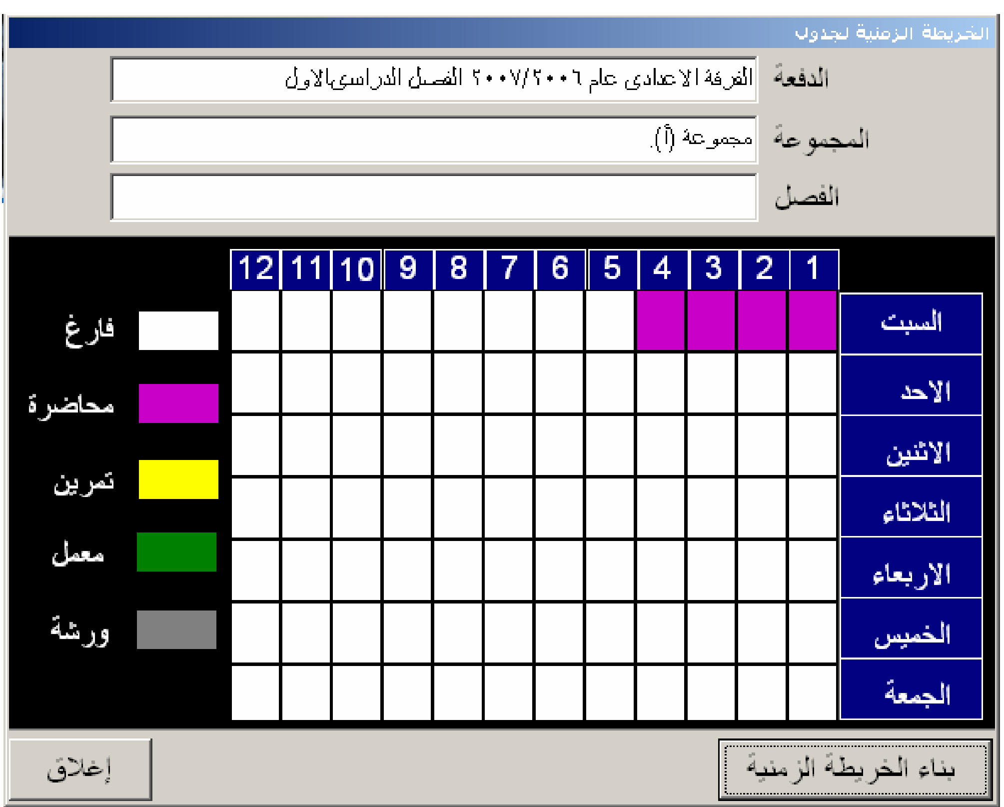
"فنيات تصميم الدالة التى سبق التعرض لها فى البرمجة الهيكلية – يقابلها هنا تصميم Method داخل الفصيلة Class "

مثال يوضح مفهوم انشاء الكائنات من Class (Instantiation)

فى هـذا المثـال لـدينا نمـوذج يقـوم برسـم الجـدول (جـدول الحصـص الدراسـية) وذلـك بانشاء مجموعة كبيرة من الكائنات جميعها من النوع SHAPE ثم بعد ذلك يقوم بتلوينها تبعا لنوع الحصة.



شكل(١٠) – (أ) النموذج قبل رسم الجدول



شكل(١٠) – (ب) النموذج بعد رسم الجدول

وفيما يلى الكود الذى يتم تنفيذه لرسم الجدول – قد تم كتابته باستخدام لغة Visual FoxPro

```
LOCAL x,y,myrec
IF thisform.tableshow = .f.
thisform.tableshow = .t.
ELSE
    FOR x = 1 TO 84
        myname = "myshape" + ALLTRIM(STR(x))
        thisform.container1.RemoveObject(myname)
    NEXT
ENDIF
If  EMPTY(thisform.text1.Value)   .or.;
    EMPTY(thisform.text2.Value)
    MESSAGEBOX("فضلا ادخل اسم الدفعة والمجموعة",0,"عفوا")
    RETURN
ENDIF
PUBLIC myobjs[84]
FOR x = 1 TO 84
        myobjs[x] = ""
NEXT
PUBLIC maxobjnum
maxobjnum = 0
FOR x = 1 TO 7
  FOR y = 1 TO 12
    maxobjnum = maxobjnum + 1
    myobjs[maxobjnum]="myshape" + ALLTRIM(STR(maxobjnum))
    myname = myobjs[maxobjnum]
    ThisForm.Container1.AddObject(myname,"shape")
    myname = "ThisForm.Container1." + myname + "."
    &myname.top = (x-1)*37
    &myname.left = (y-1)*30
    &myname.width = 30
    &myname.height = 37
    &myname.backcolor = RGB(255,255,255)
    &myname.visible = .t.
  NEXT
NEXT
SELECT mytabel
myrec = RECNO()
```

```
SET FILTER  TO  ALLTRIM(tgroupname)  =;
    ALLTRIM(thisform.text2.Value)
GOTO top
DO WHILE .not. EOF()
        SELECT mat
        LOCATE FOR mcode = mytabel->tsubcode
        IF .not. mnum = VAL(thisform.text4.Value)
            SELECT mytabel
            SKIP 1
            LOOP
          ENDIF
        SELECT mytabel
        IF tfrom = 0 .or. tto = 0
             SKIP 1
             LOOP
        ENDIF
        IF ALLTRIM(tday) = "السبت"
        myrow = 0
        ENDIF
        IF ALLTRIM(tday) = "الاحد"
        myrow = 1
        ENDIF
        IF ALLTRIM(tday) = "الاثنين"
        myrow = 2
        ENDIF
        IF ALLTRIM(tday) = "الثلاثاء"
        myrow = 3
        ENDIF
          IF ALLTRIM(tday) = "الاربعاء"
        myrow = 4
        ENDIF
        IF ALLTRIM(tday) = "الخميس"
        myrow = 5
        ENDIF
        IF ALLTRIM(tday) = "الجمعة"
        myrow = 6
        ENDIF
        FOR x = tfrom TO tto
        v = 12 - x + 1
        mypos = (myrow * 12 ) + v
        myname = "myshape" + ALLTRIM(STR(mypos))
          myname = "ThisForm.Container1." + myname + "."
          IF .not. EMPTY(tclassname)
            IF .not. EMPTY(thisform.text3.Value)
```

```
    IF ALLTRIM(tclassname)==ALLTRIM(thisform.text3.Value)
        IF TCLASSTYPE = 1
            &myname.backcolor = RGB(255,255,0)
        ENDIF
        IF TCLASSTYPE = 2
        &myname.backcolor= THISFORM.SHAPE24.BACKCOLOR
        ENDIF
        IF TCLASSTYPE = 3
        &myname.backcolor = THISFORM.SHAPE25.BACKCOLOR
        ENDIF
    ENDIF
    ENDIF
    ELSE
            &myname.backcolor = RGB(200,0,200)
   ENDIF
NEXT
        SKIP 1
ENDDO
        SET FILTER TO
        GOTO top
        GOTO myrec
        SELECT pw
        RELEASE myobjs
        RELEASE maxobjnum
```

ملحوظة هامة

فى المثال السابق نظرا لانه تم انشاء العديد من الكائنات فانه من غير المنطقى ان نعطى اسم مميز لكل واحد منهم على حده بمعنى اننا لا نسمى اسماء معرفة مثل (خسن و على) وانما نعطى اكواد مثل (١ و ٢ و ٣) وذلك باننا قمنا بانشاء مصفوفة تحتوى على الكائنات ومن ثم نتعامل مع الكائنات من خلال عناصر المصفوفة.

ايضا لعلنا نتذكر الايام الصعبة فى عصر البرمجة الهيكلية – حيث كنا نشى مصفوفة ذات ابعاد مختلفة لتخزين البيانات وكنا نواجه صعوبة فى الوصول للبيانات من قبل الدوال لانها لا ترتبط معا فى بناء واحد مثلما نجد هنا حيث نرى Class من النوع Shape وتشمل بيانات مرتبطة بالدوال التى تعمل عليها.

مثال على التركيب والتداخل Composition :-



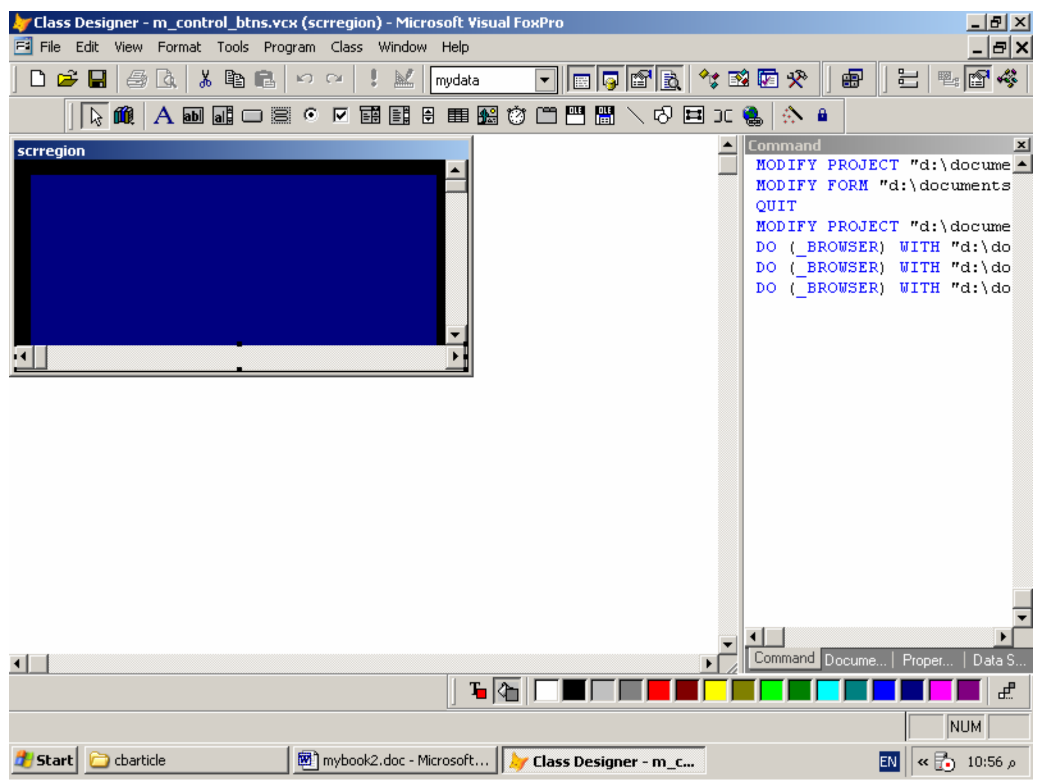
شكل(١١) : برنامج يوضح مفهوم Composition فى برمجة الكائنات

فى شكل(١١) نجد برنامج ترتكز واجهته عى Scrolling Region اى يوجد اكثر من كائن من Scrolling Region Class

وهذه ال Class تشتمل بيانات + دوال ومن ضمن هذه البيانات يوجد ٣ بيانات تمثل ٣ كائنات هما Vertical Scrollbar Object

واخر لـ horizontal scroll bar object والثالث عبارة عن Frame

وفيما يلى شكل ال Class داخل فيجوال فوكس برو – وهى من النوع Visual Class – مبنية على التصميم وليس الكود

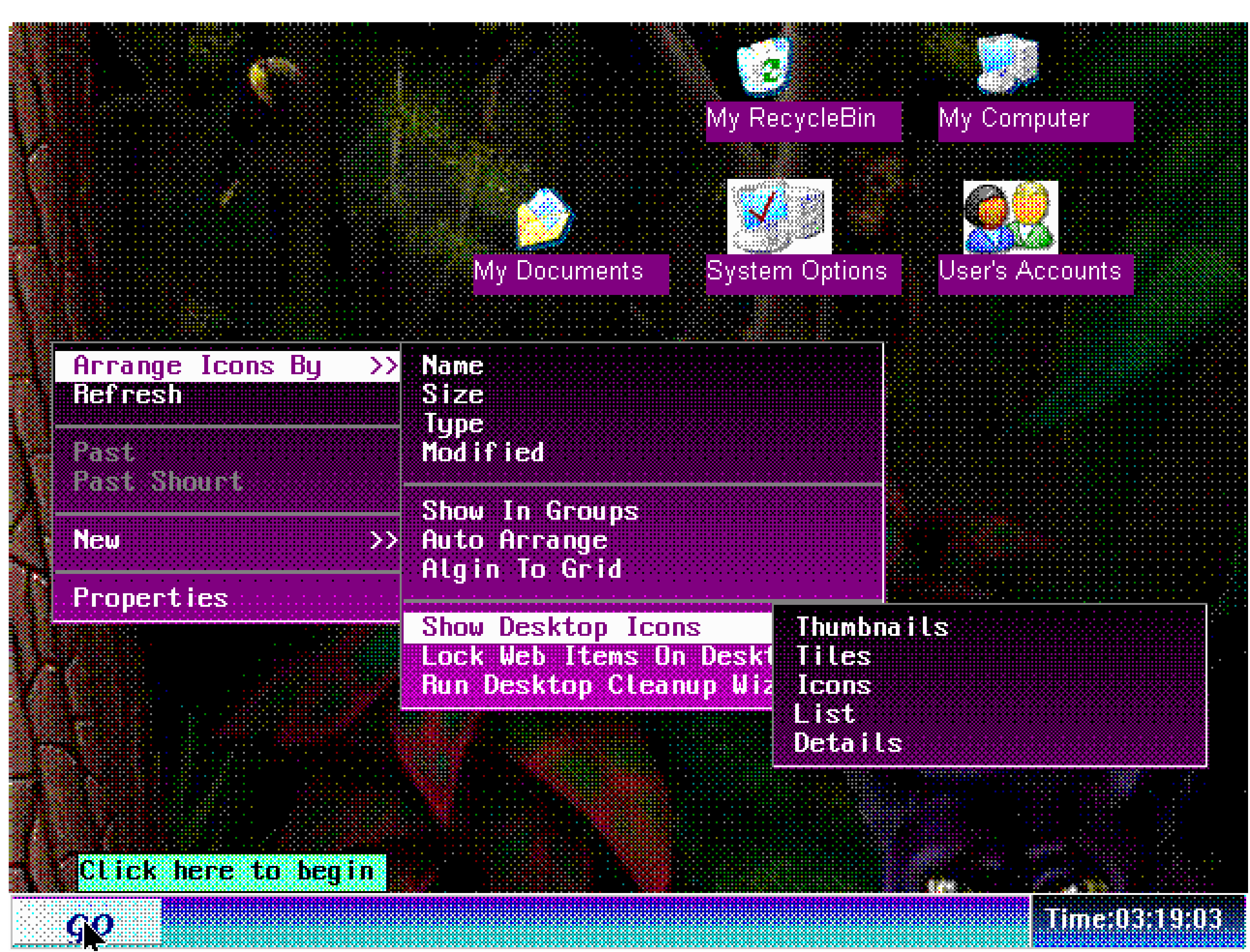


شكل(١٢) الفصيلة المرئية Visual Class

مثال على الوراثة Inheritance :-

انظر شكل(١٣) والذى يوضح قائمة Short Menu تشمل قائمة فرعية وهذه القائمة الفرعية Sub Menu تشمل قائمة اخرى فرعية

من هنا يمكن تمثيل القائمة الرئيسية ب كائن من Class اساسية Main/Mother Class ام القائمة الفرعية فهى كائن من خلال Class فرعية Sub Class متولدة من الـ Class الرئيسية والفرق بينهما ان ال Class Mother تهتم باحداثها والقوائم المتفرعة منها اما ال Class الفرعية فهى تهتم بالاحداث الخاصة بها وبالقوائم الفرعية كما انها تهتم بالقوائم الاساسية التى سبقتها.



شكل(١٣) برمجة القوائم ذات المستويات المتعددة باستخدام مفهوم الوراثة

ملحوظة هامة

هذا المثال من مشروع GUI Package الذى قام بعمله المولف – يمكن الحصول على نسختك المجانية من هذا المشروع
من خلال الموقع http://www.sourceforge.net/projects/fglib
اسم الملف FGLGUI3.ZIP

كيفية ابتكار نمط برمجة جديد !

ان نمط البرمجة هو المحرك الاساسى لعلم تصميم البرمجيات – وابتكار نمط برمجة جديد ليس بالامر اليسير وانتشار النمط الذى تبتكره وقبوله على المستوى العلمى والتكنولوجى ايضا ليس سهلا.

قبل ان تفكر فى ابتكار نمط برمجة جديد عليك ان تجيب على الاسئلة الاتية

١ – هل انا سعيد بانماط البرمجة المتاحة حاليا ؟
٢ – هل هناك نقض فى الانماط المتاحة ؟ ما هو ؟
٣ – ماذا تقترح لحل المشاكل التى تواجهها فى نمط البرمجة ؟

فى الواقع من الصعب جدا ان تنبتكر نمط برمجة جديد الا اذا حددت ماذا تريد من هذا النمط وكيف سيقدم البديل الامثل للانماط المتوفرة حاليا

المبرمجين العباقرة الذين تخطوا مرحلة الاحتراف يقومون بابتكار لغات برمجة جديدة تكون مبنية على Programming Paradigm معروف مثل Structure Programming و Object Oriented Programming ويقومون بعمل اضافات على هذه الانماط الاساسية بحيث تشمل اللغة ما يسمى ب Programming Style مثل C Programming Style و VB Programming Style وهكذا

فى الواقع ياتى نمط البرمجة لحل مشاكل جوهرية – باسلوب فكرى موحد – ان نمط البرمجة Programming Paradigm هو اسلوب تفكير يتمثل فى طريقة التصميم.

البرمجة الهيكلية قدمت حلا للبرمجة التقليدية من حيث التنظيم وسهولة المتابعة والصيانة – برمجة الكائنات قدمت حلا اكثر شمولية من حيث انتاجية برمجيات يعاد استحدامها وتطويرها بسهولة وتميزت برمجة الكائنات فى البيئة الرسومية

ولكن هل تكفى برمجة الكائنات لمتطلبات تطبيقات العصر ؟؟؟

للاجابة المثلى على هذا السوال ينبغى تذكر شى تم الاشارة عليه سابقا ان نمط البرمجة يتعلق بالكيف؟ وليس بماذا ؟
برمجة الكائنات ليست مثالية فى تطبيقات الزبون-المزود او الخادم Client-Server لان ذلك يتم تحديده اثناء العمل اى من خلال التكنولوجيا وليس من خلال التصميم Programming Paradigm فالفصيلة Class لا تبين اذا كانت Client او Server اثناء التصميم وال Method لا تحدد اذا كانت Service ام لا اثناء التصميم
كذلك ان برمجة الكائنات لاتاخذ فى الاعتبار نظام ادارة الاحداث Event-Driven System فال Method لايحدد اذا كانت Event ام لا وكذلك برمجة الكائنات لا تاخذ فى الاعتبار Complex Data Structure بل تتركه بعيدا عن التفكير على الرغم من اننا فى عصر التركيبات المعقدة من البيانات والا لما ظهرت لغة التراسل القياسية XML الشائعة الاستخدام

نمط البرمجة من المفترض ان تبنى عليه التكنولوجيا لانه وسيلة التصميم – لكن اذا زادت التكنولوجيا عن الحد وتخطت مفهوم نمط البرمجة فمن الافضل ان يعاد تصميم نمط البرمجة ليشمل تعقيدات التكنولوجيا فيصبح لدينا تصميم غنى وقوى وتتلاشى الحاجة الى تعلم تكنولوجيات التى يمكن اخفائها خلف نمط البرمجة.

وسوف نستعرض الان – نمط البرمجة الجديد (الدبل اس = الخادم الخارق او الممتاز) DoubleS (SS = Super Server)

ماذا يقدم نمط البرمجة الجديد DoubleS :-

- محاكاة لنمط برمجة الكائنات تتيح امكانيات افضل من الامكانيات المتاحة من قبل برمجة الكائنات بصورة مباشرة
  - تجميع اكثر More Capsulation
  - تقلل الحاجة الى الوراثة بنسبة ٥٠% Reduce Needs to inheritance
  - يتيح انشاء العديد من الفصائل Classes الشبه متشابهة بسهولة Working with Simi similar classes
  - يتيح ادارة الفصائل Classes بصورة مشابهة لـ الكائنات Objects اثناء وقت التشغيل Runtime
    - انشاء وحذف وتعديل (الفضائل وسماتها) فى Runtime
- ياخذ فى الاعتبار هياكل البيانات المعقدة Complex Data Structure ويقدم حلول متطورة للتعامل معها.
- ياخذ فى الاعتبار نظام ادارة الاحداث Event-Driven System اثناء فترة التصميم
- ياخذ فى الاعتبار طبيعة تطبيقات الزبون – المزود Client-Server
- ياخذ فى الاعتبار تصميم النظم System Design التى بحاجة الى السيطرة عى موارد النظام Control on System
- ياخذ فى الاعتبار التطبيقات الموزعة Distributed Applications
- ياخذ فى الاعتبار النظم المختبئة او المدفونة Embedded Systems
- ياخذ فى الاعتبار تطبيقات الشبكات المتطورة مثل Grid Computing
- نتيجة لما يقدمه هذا النمط فان النظم المعقدة عند تصميمها به يتلاشى التعقيد وتتسم بالوضوح Clear Design
- يتسم بالتنظيم Organization
- ياخذ فى الاعتبار السرية Security
- ياخذ فى الاعتبار تغير طبيعة بيئة العمل Mobility
- يقدم نظم قابلة لاعادة الاستخدام بصورة عالية جدا Reusability
- يدعم تطبيقات N-Tier بصورة هائلة فطبيعة تنظيم هذا النمط تتيح تحويل Monolith الى N-Tier بسهولة
- يخفى الحاجة الى تعلم العديد من التكنولوجيات – فقط ينبغى ان تفهم نمط البرمجة المتطور.

خريطة تعلم تصميم وتطوير النظم باستخدام DoubleS :-

1) بيئة العمل Environment الازمة
2) مفهوم النظم الموجه System Oriented
3) محاكاة علم و نظم الشبكات Networks System Simulation
4) وحدة بناء النظام Server
5) انواع الـخادم المختلفة Server Types
6) مكونات الخادم Server Units
7) مفهوم وحدة البيانات Data Unit Concept
8) محاكاة علم الكيمياء وتركيب الذرة Chemical System Simulation
9) نظام ادارة قاعدة البيانات التخيلى Virtual Database Management System
10) مفهوم وحدة التعليمات او الاكواد Code Unit Concept
11) محاكاة علم الدوائر الكهربية Electricity Circuits Simulation
12) مفهوم وحدة النقض\التصويت\التراسل Veto Unit
13) محاكاة مفهوم التفاعل الانسانى Human Interaction Simulation
14) مغهوم جمل المقاومات Resistance Statements
15) مقهوم محاكاة برمجة الكائنات Object Oriented Simulation
16) تفاصيل استخدام وحدة البيانات Data Unit Details
17) تفاصيل استخدام وحدة الاكواد Code Unit Details
18) تفاصيل استخدام وحدة النفض Veto Unit Details
19) مثال بسيط على وحدة البيانات Data Unit Example
20) مثال بسيط على وحدة التعليمات Code Unit Example
21) مثال بسيط على وحدة النقض Veto Unit Example
22) تفاصيل محاكاة برمجة الكائنات
23) مثال على خادم الجرافك Graphic Server وخادم الصوت Sound Server
24) تطبيقات الطبقات المتعددة N-Tier Applications
25) مفهوم الخادم كمترجم Server As Compiler
26) ماذا بعد الدبل اس (عالم البرمجة بدون اكواد) و Goal Designer
27) DoubleS كقاعدة للعديد من الابحاث العلمية
28) DoubleS كبنية اساسية للغات البرمجة المتطورة
29) كيفية المشاركة فى هذه الثورة العلمية

بيئة العمل الازمة :-

من المفترض ان يكون نمط البرمجة جزء لايتجزأ من مكونات اللغة – لكن نمط البرمجة DoubleS يعد نمط برمجة جديد لهذا لا يمكن ان تجده مباشرة فى اى لغة برمجة بل انت بحاجة الى اضافة بعض المكونات الى بيئة التطوير حتى تستطيع استخدام نمط البرمجة DoubleS وهذه المكونات هى

- محيط التطوير DoubleS Framework
- مكتبة DoubleS اى DoubleS Library
- مترجم متوافق مع DoubleS Library مثل Clipper او ++xBase او xHarbour او xHarbour/MiniGUI

اولا : من خلال محيط التطوير يتم تصميم النظام وكتابة التعليمات اللازمة وفى النهاية يتم ترجمة النظام الى شفيرة قياسية مفهومة DoubleS Syntax وهى عبارة عن ملفات نصية Text File يمكن التعامل مع محتوياتها باى محرر للاكواد Code Editor او Notepad الخاصة ب Microsoft Windows

ثانيا : يتم ترجمة الشفيرة الخاصة بـ DoubleS بمساعدة ملف Header يشمل تعريف ويكون المسئول عن الترجم جزء من المترجم يدعى Preprocessor والذى يحول الشفيرة الى دوال Functions تم تعريفها فى DoubleS Library ثم بعد ذلك يقوم المترجم باستخراج Object Code (بلغة الالة Machine Language)

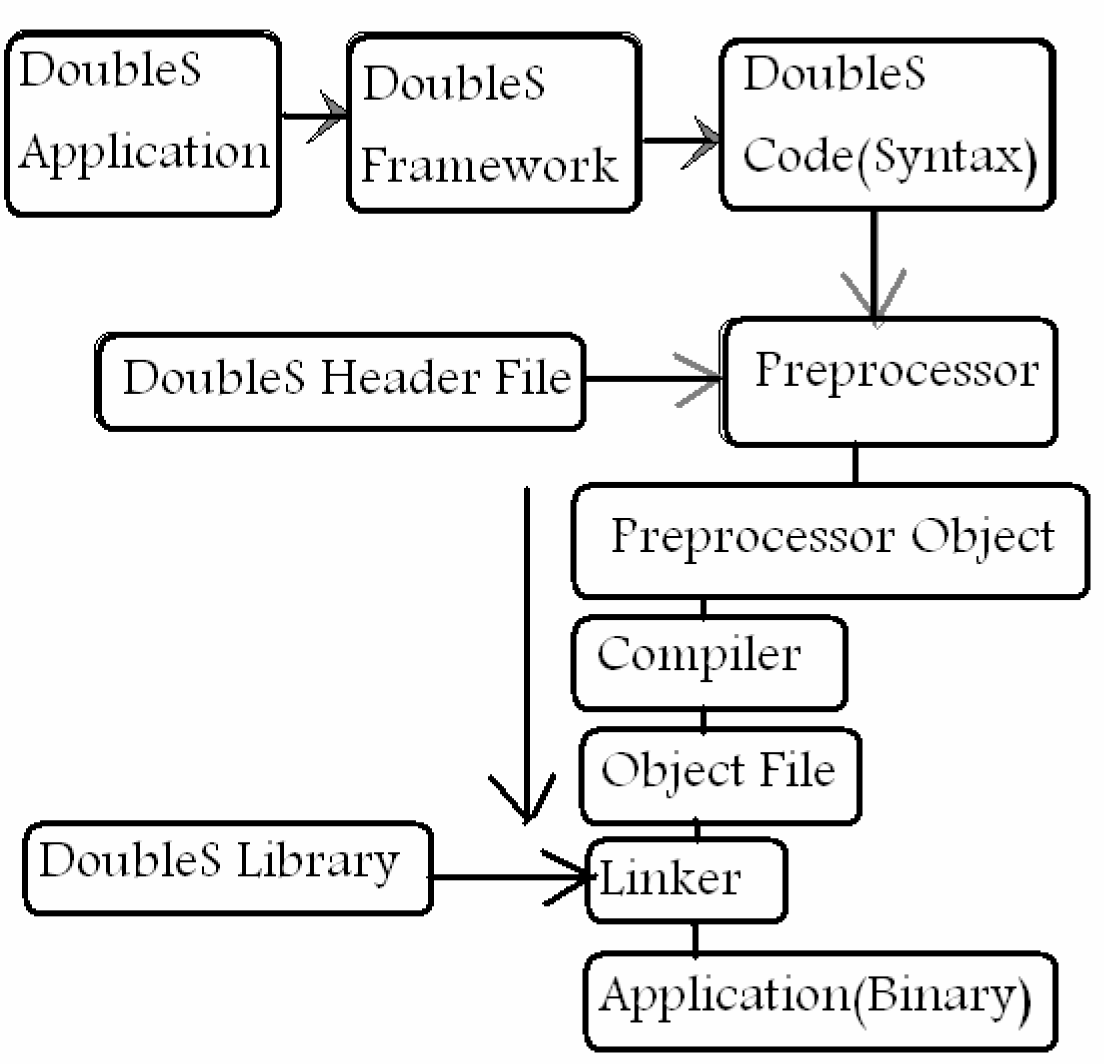
ثالثا : يمكن استخدام اى رابط Linker لاستخراج Binaries تكون متوافقة مع نظام التشغيل وهنا يلجا الرابط الى مكتبة DoubleS اى DoubleS Library حتى يستطيع تفسير محتويات Object Code الى Executable Code يستطيع نظام التشغيل التعامل معه.

انظر شكل (١٤) والذى يوضح المراحل التى يمر بها النظام المطور باستخدام DoubleS

ملحوظة هامة

ان المترجم الذى ارشحه للعمل مع DoubleS هى xHarbour/MiniGUI والذى يتيح عمل تطبيقات تعمل بالبيئة الرسومية GUI تحت نظام Microsoft Windows وهو مجانى يمكن الحصول عليه من الموقع
http://www.sourceforge.net/projects/harbourminigui

يمكن استخدام المترجم xHarbour فقط مع DoubleS وهو مترجم Portable يتوفر لاكثر من نظام تشغيل مثل Dos و Windows و Linux و OS/2 وغيرها من الانظمة وهو مجانى http://www.sourceforge.net/projects/xharbour



شكل(١٤) بيئة العمل ومراحل تطوير النظام باستخدام DoubleS

كما سبق الاشارة فان المترجم الذى سوف تحتاجه – مجانى ومن السهل الحصول عليه ايضا فان بقية مكونات بيئة العمل مجانية ويمكن الحصول عليها من
http://www.sourceforge.net/projects/doublesvsoop
ومن هذا الرابط يمكنك الحصول على نسختك من محيط التطوير DoubleS Framework والمكتبة DoubleS Library كما يمكنك الحصول على العديد من المصادر الاخرى اللازمة لتعلم واستخدام DoubleS

مفهوم النظم الموجهة :-

يضع نمط البرمجة DoubleS فى الاعتبار انه Software الذى يتم تطويره هو عبارة عن نظام System والذى يتطلب عدة معاير قد لا نحتاج لاخذها فى الاعتبار عن تطوير البرامج Programs او التطبيقات Applications وعلى الرغم من ان نمط البرمجة DoubleS يمكن استخدامه فى تطوير البرامج او التطبيقات البسيطة الا انه دائما يطرح المفاهيم التى نحتاج اليها فى النظم المعقدة.

ليس هذا فقط بل يرتكز نمط البرمجة DoubleS على تحرير النظم من الجمود ويضيف اليها مستويات مختلفة من المرونة مما يسمح باندماج الانظمة فى المستقبل او تعاون النظم المختلفة معا

يطرح نمط البرمجة DoubleS مفهوم مراقبة النظام اثناء العمل مما يسمح بادارة موارد النظام المادية من قبل النظام فسه بدون الاعتماد الكلى على لغة البرمجة او النظام التشغيل.

ان من اهم مايميز نمط البرمجة DoubleS هو تعدد مستويات النظام – فقد تكون هناك نظم بسيطة وعندها ترتكز على خادم واحد One Server (الخادم Server هو وحدة بناء الـ DoubleS – مثلما نجد الفصيلة Class وحدة بناء OOP) وقد
يكون النظام عبارة عن مجموعة من الخوادم Group Of Servers وقد نضعها معا تحت سيطرة خوادم اخرى للادارة بحيث نكون مايسمى بالـ DoubleS System والذى يتم فيه تنفيذ اهداف النظام ومراقبت كيفية التنفيذ من قبل النظام لنفسه فى نفس الوقت).

ملحوظة : عند تجمع مجموعة من الخوادم معا فانها تكون مايسمى بالشبكة Network وهى مستوى اخر من مستويات النظام

محاكاة علم الشبكات Networks Systems Simulation :-

ان ابتكار الحاسب يعد ثورة علمية وابتكار الشبكات هى الثورة الثانية ولكن هذه الثورة وما نتج عنه من الحاجة الى تطور فى البرمجيات ادى الى حدوث شى من التعقيد فبدلا من تطبيقات Desktop الى كانت تعمل على جهاز واحد – ظهر ما يسمى بـ
Server Based Applications مثل انظمة قواعد البيانات التى تعمل على الخادم ثم تطور المفهوم لنرى Client-Server Applications والذى يرتكز على شطر النظام الى نصفين – جزء مسئول عن الطلب والبحث عن خدمات والجزء الاخر مسئول عن
تلبية هذه الخدمات – ولم يقف الامر عند ذلك بل امتد ليواكب التطور فى علم الشبكات وظهور شبكة الانترنت العالمية ومن هنا ظهرت تطبيقات الويب Web Applications والتى ترتكز على الانترنت وظهر ايضا مايسمى بخدمات الويب Web Service وهى عبارة عن كائنات Objects من فصائل Classes ولكن يتم نشرها عبر الانترنت Web Publishing ويمكن لصفحات الويب ان تستخدمها – كما انه يمكن استخدمها من قبل برمجيات Clients لها واجهة مستقلة User Interface لاترتكز على متصفح الانترنت.

هذا هو الواقع الذى نعاصره ومع ذلك يوجد نظرة نحو المستقبل حيث ان علم الشبكات الان يتجه نحو الجيل الثانى من الانترنت

وهو مايعرف بـ Internet 2 والذى يفترض ان يـدعم الجيـل الجديـد مـن التطبيقـات وهـو مايعرف بـ Grid Computing والذى يعتمد على استخدام كم كبير من الاجهزة المتصـلة بالشبكة كجهاز واحد Super Computer تخيلى – يمكن من خلال ذلك اداء العديد من العمليات المعقدة التى قد تستهلك زمنا طويلا جدا – فى فترة قصيرة.

ان Grid Computing يرتكـز علـى مـا يسـمى ب Grid Information Service والـذى سوف يحل محل Internet Information Service وهنا فى الـ Grid Computing يكـون مسئولا عن استقبال Request المعقد من Application الذى يعمل على End System ويـتم اداء العمليـات المعقـدة مـن قبـل الخـوادم الموجـودة بالشـبكة والتـى تكـون Intermediate System وحتى يحـدث ذلـك فاننـا نتعـرض لعمليـة Procasting اى توزيـع المهة Job والتى تكون Dynamic وتختلف حسب الحاجة.

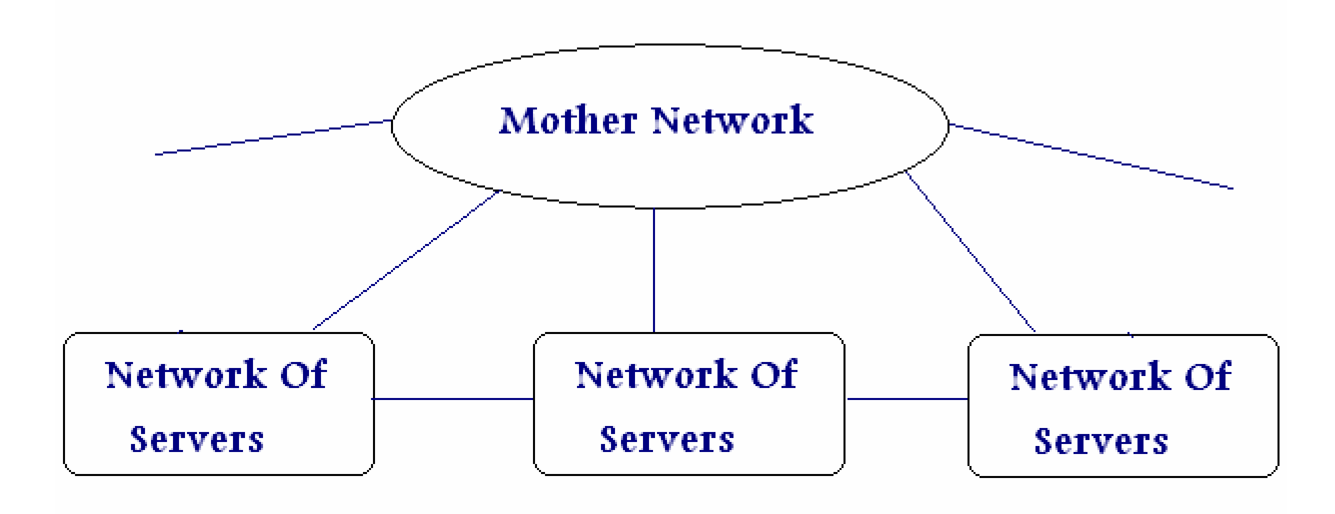
اننا فى عـالم برمجـة الشـبكات – وبرمجـة الكائنـات رغـم روعتهـا الا انهـا لاتعـدو كونهـا وسيلة جيدة كانـت ملائمـة جـدا لكـن فـى عـالم قـد والـذى كـان يعـرف بعصـر البيئـة الرسومية GUI – وان استخدام برمجة الكائنا بالوضع الذى هى عليه فى تطـوير الـنظم الحديثة يعد نوع من السكون والذى يرفضه العلماء والبـاحثين فـى مجـال نمـط البرمجـة Programming Paradigm.

ان نمط البرمجة DoubleS يحاكى التطور المذهل فى الشـبكات وعمليـة المحاكـاة لـم تتوقف على اخذ مثل هذه النظم فى الاعتبار بل امتد لنصل الى اننـا فـى الــ DoubleS نقوم بتطوير خادم Server وليس Class.

ان النظرة العليا على عالم الـ DoubleS تظهر اننـا لـدينا مجموعـة مـن الشـبكات ويجـد على الاقل شبكة تسمى الشبكة الام او مايسمى بـ Mother Network وكـل شـبكة تشمل مجموعة من الخوادم

انظر شكل (١٥) والذى يوضح ذلك.

DoubleS Application Design



شكل(١٥) النظام مجموعة من الشبكات – تشمل مجموعة من الخوادم

ان محاكاة علم الشبكات فى نمط البرمجة تسهل عملية التصميم من الواقع – بمعنى ان Distributed System المكون من مجموعة من الشبكات و الخوادم المتصلة معا – يتم تصميمه بصورة مشابهة للواقع الذى سوف يعمل فيه.

ان عملية الشبكات والخوادم لا تعنى فرض كون النظام Distributed وانما تسهل امكانية جعل النظام Distributed ان تقسيم النظام الى شبكات وخوادم يعطى مستويات مختلفة من الدمج والترابط ولا يفرض علينا مكونات Hardware مطابقة لمواصفات التصميم.

وحدة بناء النظام Server :-

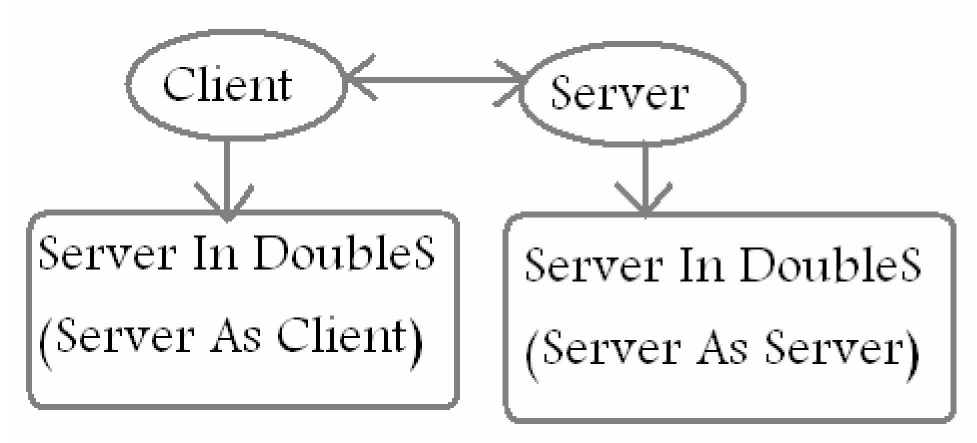
مفهوم قوى ويستدل من معناه ان هذا المكون يقدم خدمات – مما يعنى انه يضع فى الاعتبار عملية الاتصال عن بعد بالخوادم الاخرى – وهنا يظهر مفهوم هام

الخادم : قد يقدم خدمات وقد لايقدم خدمات – ومن الممكن ان يكون Client وهذا يعنى ان كلمة خادم تفيد بان هذا المكون لديه عوامل تجعل من السهل جعله يعمل كخادم ولا تشترط ذلك – كما ان هذا المكون من الممكن ان يكون زبون Client يطلب خدمات من خادم اخر.

ايضا هناك عامل هام – الخادم Server فى DoubleS من الممكن ان يقدم خدمات وهذا يعنى انه اثناء عمله وادائه للمهام التى يقوم بها فانه يجب ان يكون مستعدا لاستقبال طلبات فى اى وقت وهنا تظهر الحاجة الى Secure & Transparent Link

بين الخادم Server الذى يعطى الخدمات وبين الخادم الاخر Another Server الذى يطلب الخدمة وهنا يطلق عليه Server As Client

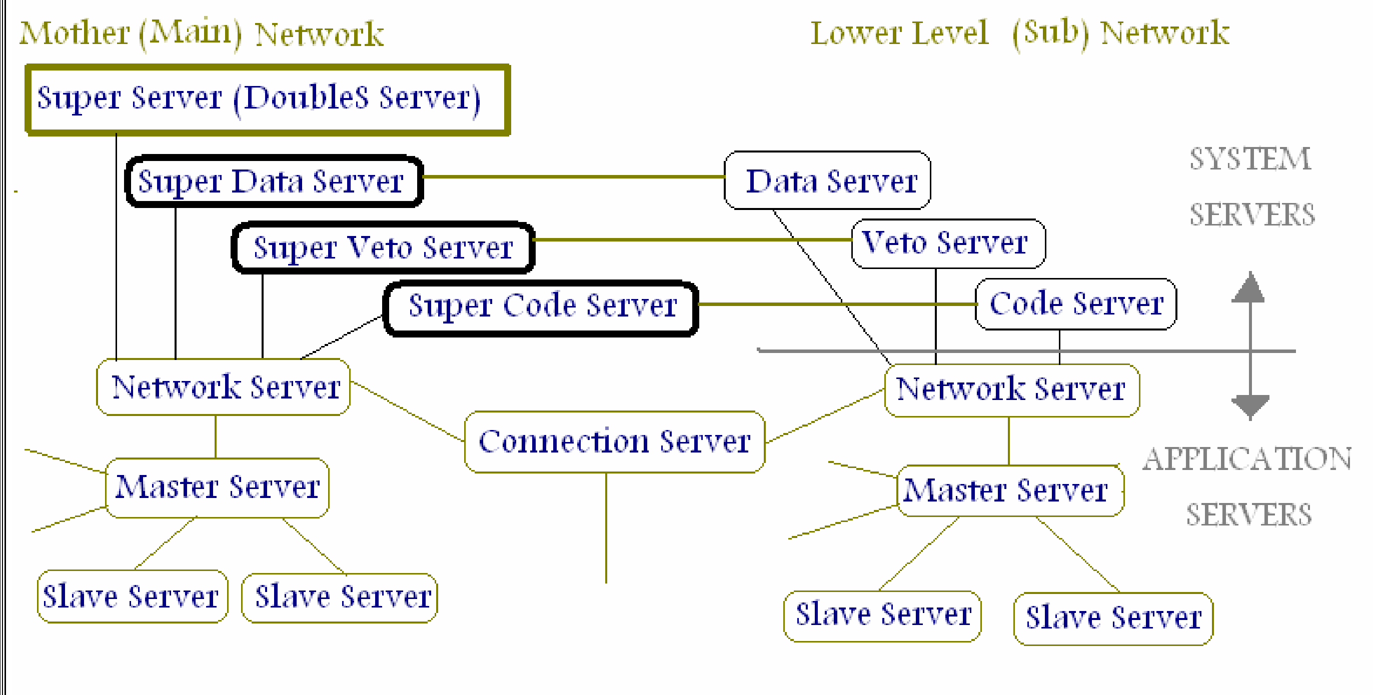
انظر شكل (١٦) والذى يوضح ذلك المفهوم.



شكل(١٦) كفهوم الخادم كمزود و كعميل.

انواع الخادم المختلفة Server Types :-

هناك خوادم عامة لاى غرض وهناك خوادم محددة الغرض وذلك امر بديهى – وسبق الاشارة ان DoubleS يدعم مفهوم النظم الموجهة System Oriented ومن هنا تم فرض ان النظام عبارة عن مجموعة من الشبكات بها مجموعة من الخوادم ولكن تم تقسيم الخوادم الى نوعين – نوع يودى الغرض من النظام ونوع يقوم بمراقبة النظام.
انظر شكل (١٧) والذى يوضح انواع الخادم المختلفة



شكل(١٧) انوع الخادم المختلفة داخل DoubleS System

- فى الواقع هناك ١٣ نوع من انواع الخادم
- Procedure Server (only code unit)

الخادم الذى يشمل وحدة تعليمات فقط

- Class Server (without veto unit)

الخادم الذى يشمل وحدة تعليمات + وحدة بيانات فقط

- Network Server
- Connection Server
- Master Server
- Slave Server
- Data Server
- Code Server
- Veto Server
- Super Data Server
- Super Code Server
- Super Veto Server
- Super Server (Doubles Server)

وسوف نتعرف الان على وظائف كل نوع من هذه الانواع :-

- Super Server: the server which plays the role of the God in the system and can control all the DoubleS system from start to end.

الخادم الذى يلعب دور الاله فى النظام ويمكنه التحكم بكل شى من البداية حتى النهاية داخل النظام ككل

- Super Data Server: the server which control the data system in the DoubleS application.

الخادم المسئول عن التحكم بوحدة البيانات الخاصة بالنظام

- Super Code Server: the server which control the code system in the DoubleS application.

الخادم المسئول عن التحكم بوحدة التعليمات داخل النظام

- Super Veto Server: the server which control the veto system in the DoubleS application.

الخادم المسئول عن التحكم بوحدة النقض والتراسل الخاصة بالنظام

- Network Server: the server which link system servers with application servers.

الخادم الذى يعرف الشبكة بين مجموعة من الخوادم

- Master server: the main server in the application servers.

الخادم الرئيسى داخل الشبكة فى خوادم التطبيق الذى نطوره

- Slave server: server which can't work alone in network without master server.

الخادم الذى لايعمل مع مكونات الشبكة بدون خادم رئيسى

- o Connection server: server which connects between networks (2 or more of networks).

الخادم المسئول عن ربط شبكتين معا

- o Data Server: Server which control data system in sub network.

الخادم المسئول عن التحكم بوحدة البيانات الخاصة بالشبكة المعرف فيها فقط وليس النظام ككل

- o Code Server: Server which control code system in sub network.

الخادم المسئول عن التحكم بوحدة التعليمات داخل الشبكة المعرف فيها فقط وليس النظام ككل

- o Veto server: Server which control veto system in sub network.

الخادم المسئول عن التحكم بوحدة النقض والتراسل داخل الشبكة المعرف فيها فقط وليش النظام ككل

ملحوظة هامة

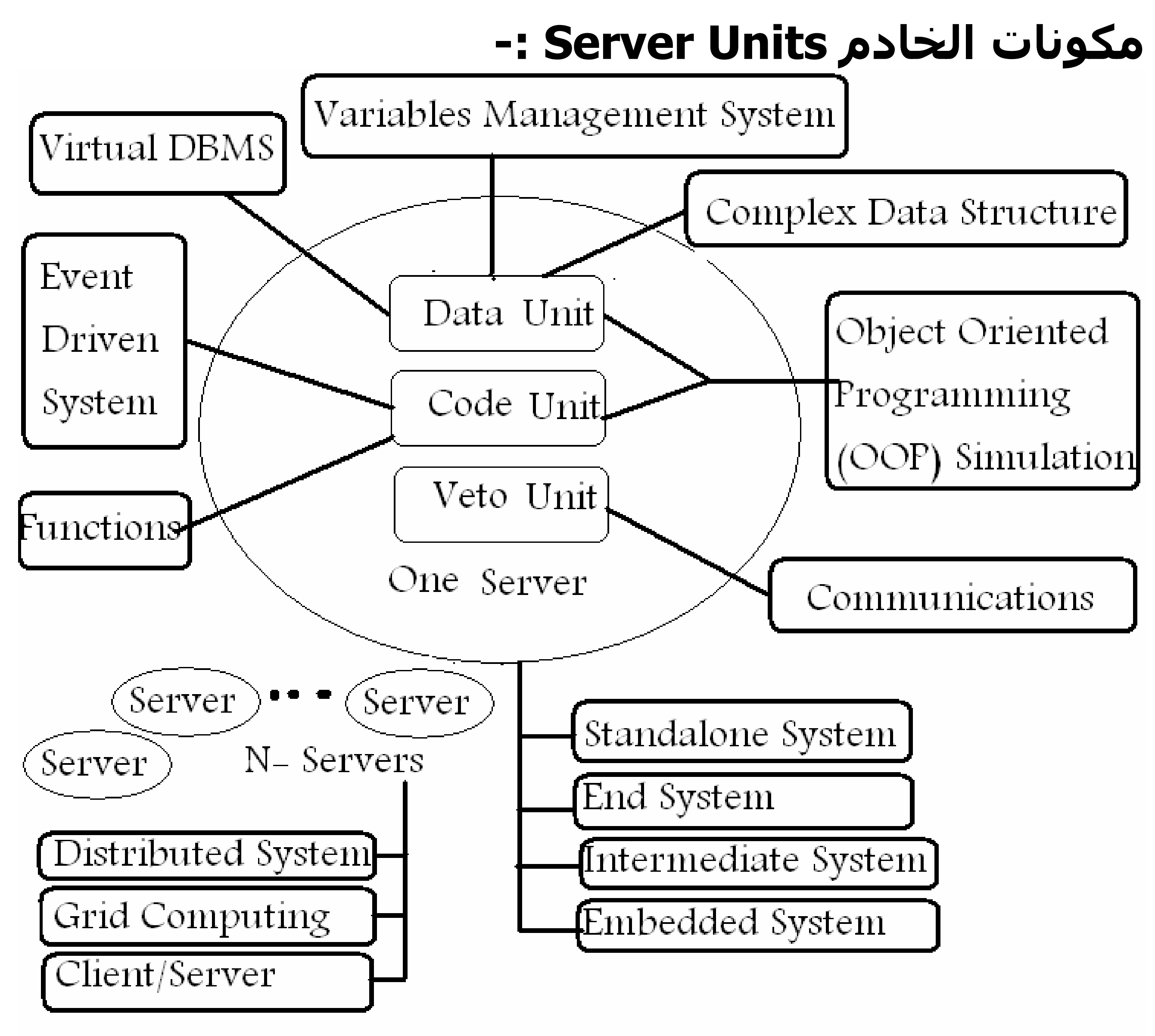
الخوادم الخاصة بالنظام مثل Super Server, Super Data Server, Super Code Server & Super Veto Server هى خوادم لايقوم المبرمج ببرمجتها من الصفر – وانما يجد لها Template ثابت يستخدمه المبرمج – حيث انها عبارة عن اعلان لاحداث Handle To Events والتى عند تحققها يتم تنفيذ الاكواد التابعة لها

بالمثل ايضا الخوادم Data Server, Code Server & Veto Server والتى يكون لها Template ثابت هى الاخرى

ينبغى التاكيد على ان مفهوم الشبكات والخوادم هنا تخيلى بمعنى ان كل خادم لايشترط ان يكون له جهاز خادم فعلى مقابل له فى الواقع وبالمثل الشبكات هنا تخيلية ولا يشترط ان يكون هناك شبكات حقيقية مقابلة لها فى الواقع.

استخدام مفهوم الشبكات والخوادم لمرونة تصميم النظم التى تحتاج بيئة من هذا النوع ولكن لا تمثل قيود على توفر مثل هذه البيئة.

مكونات الخادم Server Units :-



شكل(١٨) مكونات الخادم واستخدامات كل منها

يتكون الخادم من مجموعة من الوحدات Group of Units على الاقل – ٣ وحدات قياسية هى

١ – Data Unit اى وحدة البيانات وهى مسئولة عن :-
١ – تمثيل هياكل البيانات المعقدة Complex Data Structure
٢ – نظام لادارة المتغيرات Variable Management System
٣ – نظام تخيلى لإدارة قواعد البيانات Virtual DBMS
٢ – Code Unit اى وحدة التعليمات وهى مسئولة عن :-
١ – تمثيل الدوال او الوظائف Functions
٢ – نظام إدارة الاحداث Event Driven System
٣ – Veto Unit اى وحدة النقض والتراسل وهى مسئولة عن
عمليات Client/Server الزبون-المزود وتشمل

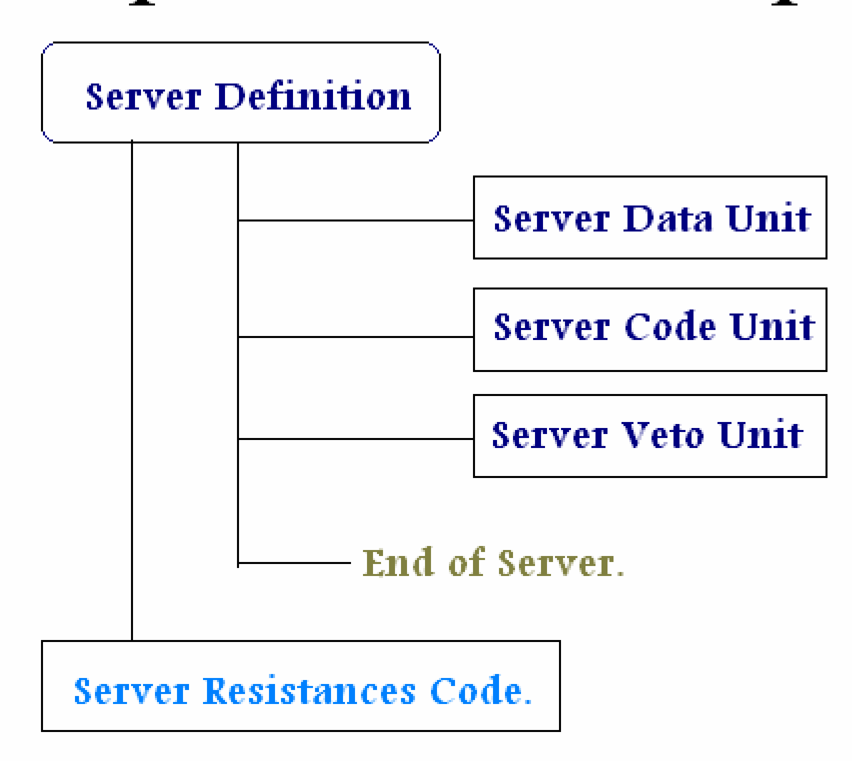
١ – استقبل بيانات
٢ – ارسال بيانات
٣ – طلب خدمات
٤ – تقديم خدامات
انظر شكل (١٨) والذى يوضح ذلك

ملحوظة هامة

نلاحظ ان الخادم بمكوناته يمثل نظام قد يكون Standalone اى نظام قائم بذاته او End اى يطلب خدمة او يقدم خدمة او Intermediate اى يكون وسيط بين نظامين او Embedded اى مختبى داخل نظام اخر.

بينما مجموعة من الخوادم معا فانها تمثل نظام زبون مزود Client-Server System او Distributed System او Grid Computing

والان دعنا ننظر الى مكونات الخادم بصفة عامة بصرف النظر عن استخدام تلك المكونات – انظر شكل (١٩)



شكل(١٩) – مكونات اى خادم Super Server

نلاحظ من خلال الرسم وجود مايسمى بـ Server Definition وهو مايقصد به عملية تعريف الخادم وذلك بتحديد ٣ اشياء هى
١ – اسم الخادم (وهو اختيارى)
٢ - نوع الخادم (سبق وذكرنا انه يوجد ١٣ نوع مختلف)
٣ – القيمة المميزة للخادم Eigen Value
وقد يتم اضافة بعض الخصائص الاخرى الازمة لاستكمال التعريف حسب نوع الخادم According to Server Type

Super Server Definition

```
New Server < Server Name > Type < Server Type > Eigen Value < Server ID >
.......specific details for each server type
```

ايضا نرى اسفل الشكل Server Resistances Code اى التعليمات الخاصة بالمقاومات Resistances التى يتم تعريفها بوحدة التعليمات او الاكواد Code Unit والجدير بالذكر التذكير بان المقاومة Resistance ماهى الا تمثيل لكل من Function و Method و Event ويتوقف النوع حسب الاستخدام.

فمثلا قد تكون المقاومة Resistance هى دالة Function عادية يتم مناداتها بصورة مباشرة او من قبل نظام النقض والتراسل Veto System على سبيل المثال عند تقديم خدمة لخادم اخر.

– او قدتكون Event اذا ما سبقت بشرط Condition وبذلك تندرج تحت نظام ادارة الاحداث Event-Driven System او قد تكون Method اذا ماتم استخدامها من خلال محاكاة برمجة الكائنات Object Oriented Simulation.

مفهوم وحدة البيانات Data Unit Concept :-

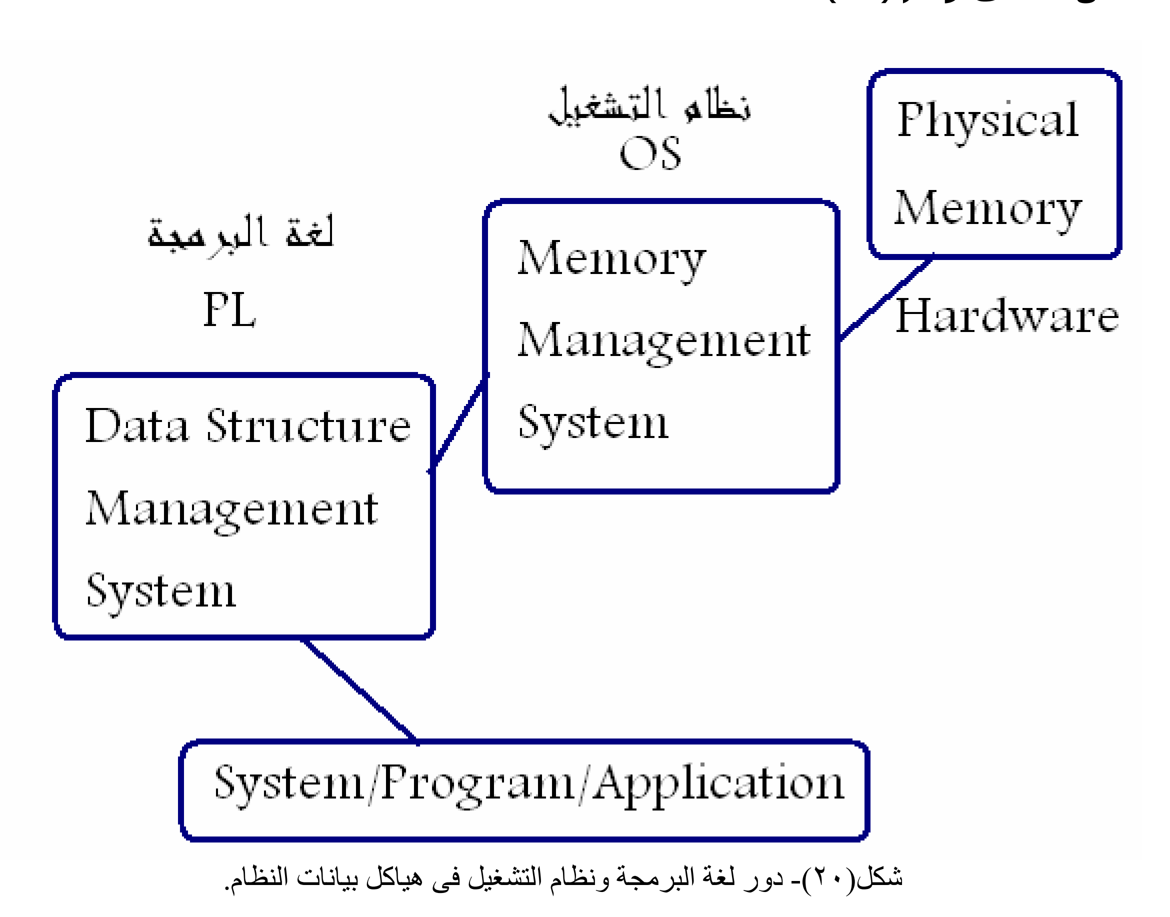
لم تعطى البرمجة الهيكلية Structure Programming اى اهتمام لهياكل البيانات الخاصة بالنظام وانما كان التركيز على الدوال بينما التفتت برمجة الكائنات لتلك النقطة – ولكن للاسف لم تقدم سوى مفهوم Encapsulation اى دمج البيانات مع الدوال معا فى الفصيلة Class ولكنها لم تعطى البيانات اهتمام اكثر من ذلك على الرغم من ان الوراثة تشمل وراثة السمات Properties التى تتضمن
البيانات Data وعلى الرغم من مفهوم التركيب Compositon الذى يستخدم Data Type وهو الكائن Object بحيث يكون احد سمات الفصيلة الا ان ذلك الاهتمام ليس كافيا وخاصة فى النظم ذات هياكل البيانات المعقدة.

ان نمط البرمجة DoubleS يقدم من خلال مفهوم وحدة البيانات Data Unit دعم كبير لهياكل البيانات Support For Data Strucutre ولا يقصد به ان يكون بديل لهياكل البيانات Data Unit is not replacement for Data Structure لان هياكل البيانات هى

مسئولية لغة البرمجة ولا ينبغى لنمط البرمجة ان يسلب ذلك It's Not the job of the programming paradigm.

يقدم نمط البرمجة DoubleS مفهوم جديد من قبل وحدة البيانات وهو ادارة هيكل البيانات الخاص بالنظام Data Structure Management وعملية الادارة تتمثل فى التعامل مع هيكل بيانات النظام كما لو كان بناء مرن يسهل تشكليله – نحن نعلم ان هياكل البيانات تنقسم الى نوعين هما Static Data Structure و Dynamic Data Structure ولا يقصد بالمرونة والادارة الخاصة بوحدة البيانات التعرض لهذين النوعين وانما يقصد وضع طبقة بين هيكل البيانات وبين موارد النظام – ولكن هذه الطبقة التى توجد بالفعل من قبل لغة البرمجة – سوف يتم استبدالها بطبقة اخرى من خلال نمط البرمجة DoubleS عن طريق وحدة البيانات والاختلاف هو ان هذه الطبقة سوف تكون طبقة متحررة تحت سيطرة المبرمج اثناء التنفيذ العملى – وتحت منظور المصمم اثناء تصميم النظام وبذلك يطغى نمط البرمجة الجديد DoubleS على الحاجز

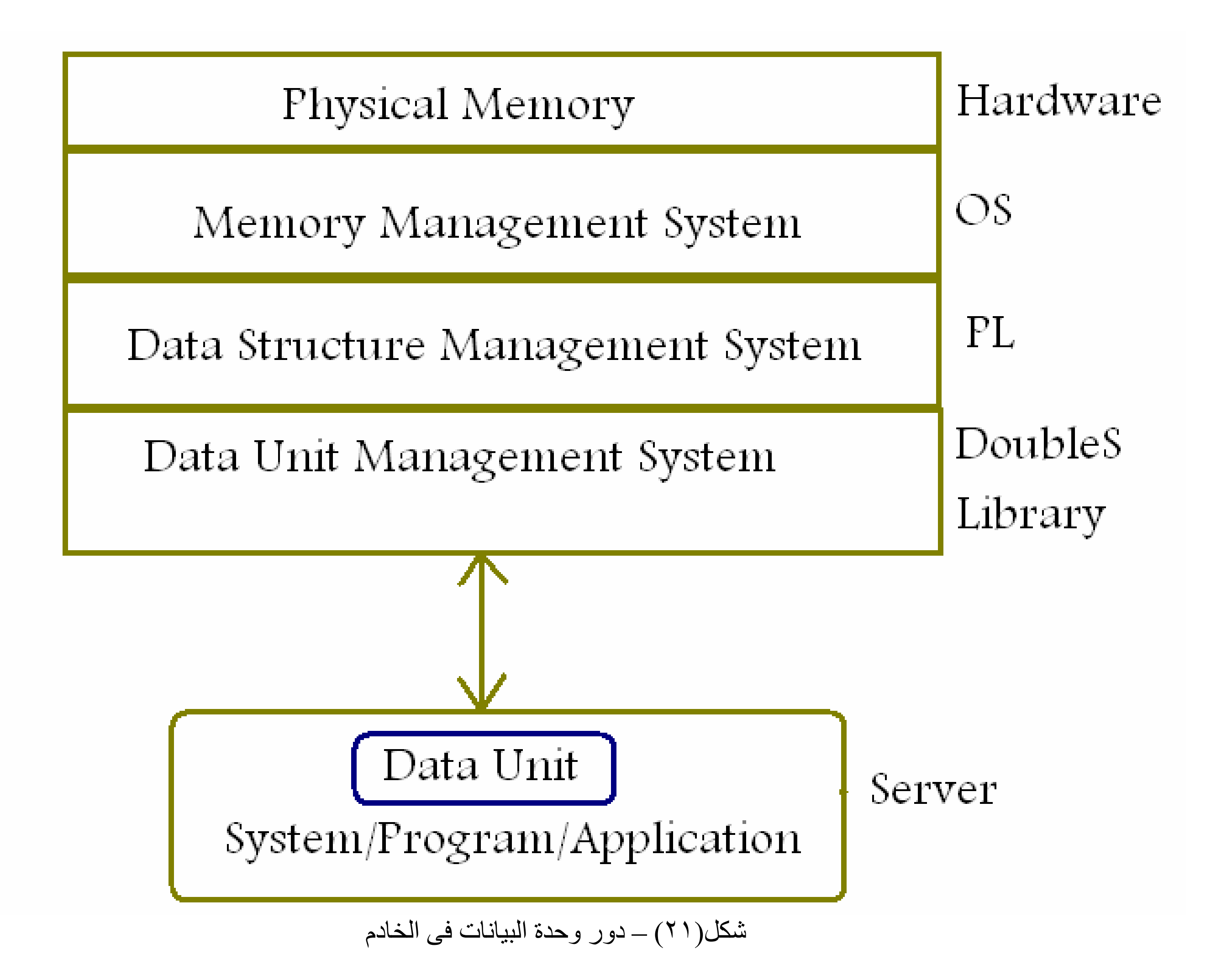
الذى تفرضه لغة البرمجة بخصوص ادارة هياكل البيانات.
انظر الشكل التالى رقم (٢٠).



شكل(٢٠)- دور لغة البرمجة ونظام التشغيل فى هياكل بيانات النظام.

اى ان وحدة البيانات هى طبقة بين النظام او البرنامج او التطبيق الذى نطوره باستخدام DoubleS وبين لغة البرمجة للتحكم بهياكل بيانات النظام. والجديد ان هذه الطبقة يمكن التحكم بها بمرونة عالية كما انها ذات مواصفات قياسية تسمح بوضوح وقوة التصميم بالاضافة الى إضافة ملامح جديدة اثناء التطبيق العملى.

ولكى يتضح ذلك المفهوم انظر شكل (٢١).



شكل(٢١) – دور وحدة البيانات فى الخادم

اما بخصوص اهداف وحدة البيانات فهى

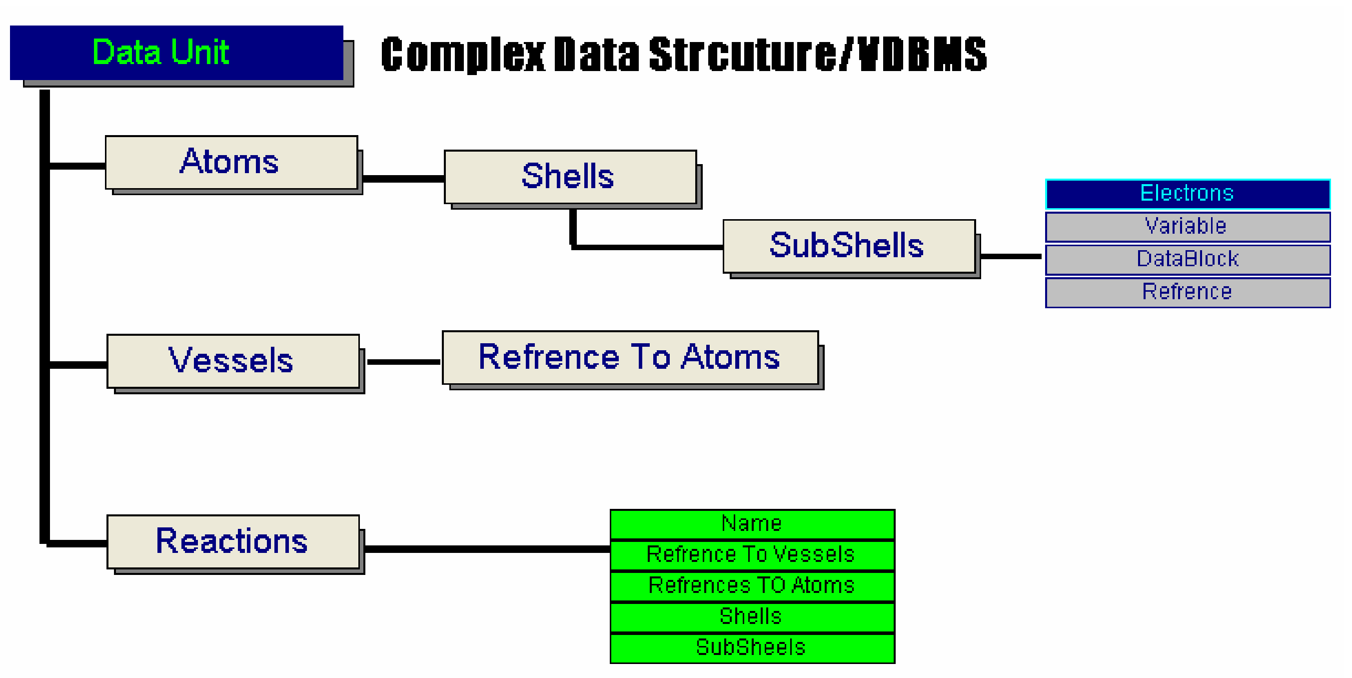
- تمثيل هياكل البيانات المعقدة Complex Data Structure بصورة منظمة
- ادارة المتغيرات Variables Management System الى ادنى مستوى مثل Physical Representation
- الحصول على ملامح جديدة مثل نظام ادارة قاعدة بيانات تخيلى Virtual DBMS
- الحصول على طبقة توضيح جديدة New Abstract Layer فى وقت التصميم Design time لم تتوفر من قبل.
- مرونة تصميم النظم التى تعتمد على تبادل كم ضخم من البيانات المنظمة بدقة عالية.

وهنا جاء الوقت لكى نشير بان النظام المستخدم فى وحدة البيانات هو نظام الكيمياء Chemical System (لانه تم استعارته من مفهوم تركيب الذرة والتفاعلات الكيميائية) وهذا النظام يعطى كفاءة عالية فى تحقيق اهداف وحدة البيانات.

ملحوظة هامة

قبل التعرض لتفاصيل هذا النظام ينبغى الاشارة الى انه محاكاة ظاهرية Simulation لنظام الذرة ولا يعنى ذلك التقيد بالقيود الطبيعية فى هذا النظام لان هذا غير مقبول عقليا – بمعنى ينبغى ان تكون المحاكاة دائما لتحقيق الفائدة وليس فى وضع القيود الغير لازمة لهذا فالتشابه مع علم الكيمياء قد يكون فى بعض المسميات والتركيبات وليس فى القيود التى تفرضها الطبيعة الا عند الحاجة.

محاكاة الكيمياء Chemical System Simulation :-



شكل(٣٢) :- تركيب وحدة البيانات المصمم على محاكاة النظام الكيميائى

انظر شكل (٣٢) والذى يوضح مكونات وحدة البيانات – انها محاكاة صريحة للنظام الكيميائى Chemical System حيث يوجد لدينا ثلاثة عناصر اساسية وهى :-

١ – الذرة Atom (مجموعة من المدارات وبالتحديد ٧ مدارات مجتمعة معا)
المدار (مجموعة من المدارات الفرعية وبالتحديد ٤ مدارات فرعية مجتمعة معا)
المدار الفرعى (يشمل مجموعة من الالكترونات)
الالكترون) وصف لنوع غير محدد من البيانات وقد تكون متغيرات)
٢ – الوعاء Vessel
يحمل عناوين مجموعة من الذرات لكى يشير إليها
٣ – التفاعلات Reactions
عبارة عن تصفية ولكن للالكترونات

- Atom : Group of shells (7 Shells) packed together

- Shell : Sub Title from atom contain group of sub shells packed together(4 Shells)
- Electrons: symbol refers to unknown type of data (memory variable, resistance(Method or Function), or arguments(Parameters))
- Vessel : something like array but it's elements are only references for atoms
- Reaction: something like filter but for electrons (filtering through shells or sub shells for atoms in vessels)

والسوال الان : لماذا محاكاة علم الكيمياء ؟
منذ زمن بعيد حينما كنت فى المرحلة الثانوية – وكنت ادرس مادة الكيمياء سالت نفسى الاسئلة التالية :-

1. Why every thing in the world returns to atoms?
2. Why reactions change the nature of the elements?
3. Why atoms are organized to shells and sub shells?
4. Why the biggest atom contains only 7 shells?
5. Why the shell can contain up to 4 sub shells?
6. Why the electron is very important to atom?
7. How the electron can move from shell to another?
8. Why there is limited number of electrons in sub shell?

١ – لماذا كل شى فى الكون يعود الى الذرات ؟
٢ – لماذا التفاعلات تغير طبيعة العناصر ؟
٣ – لماذا تنظم الذرة الى مدارات ومدارات فرعية ؟
٤ – لماذا تشمل الذرة ٧ مدارت اساسية ؟
٥ – لماذا يشمل كل مدار اساسى ٤ مدارت فرعية ؟
٦ – لماذا يعد الالكترون مهما بالنسبة للذرة ؟
٧ – كيف يستطيع الالكترون الانتقال من مدار لاخر ؟
٨ – لماذا يكون هناك عدد محدد من الالكترونات فى المدار الفرعى ؟

وكانت الاجابة على تلك الاسئلة كالتالى

1. When every thing returns to one thing, we can make every thing by only having the thing that makes Every thing
2. The environment changes change the objects that this environment contains, so we can control Our objects through controlling the environment
3. Every thing desperate to levels and sub levels make more organization for example our grades In school (so bad, bad, good, very good & excellent)

4. Every thing should have limits, so we can see the end of that thing, or feel with it
5. if my father who gives me money have only have 100,000$ in his balance the bank , I can't have more than this balance, may be much more less
6. Because it's the simple thing that can do action
7. So the action can be movable & more effective
8. Because limits mean some stability and feeling with borders of nature.

- عندما يعود كل شى الى اصل واحد يمكننا عمل اى شى بمجرد امتلاكنا لهذا الاصل
- تغيرات البيئة تغير الكائنات التى تشملها هذه البيئة لذلك يمكننا التحكم بالكائنات من خلال التحكم بالبيئة
- عند تقسيم كل شى الى مستويات ومستويات فرعية نحصل على طبقة من التنظيم
- لابد ان يكون لكل شى حدود بحيث يمكننا معرفة هذه الحدود او الشعور بها والتعامل مها بنظام
- الحدود تعطى نوع من الاستقرار والشعور بحدود الطبيعة امر ضرورى

وبعد الاجابة على تلك الاسئلة واثناء عملى على الحاسب وجهت لنفسى السوال التالى

What if we have chemical system data structure in our programming language, is that helpful?

وبعد خمس سنوات واثناء دراستى فى الجامعة وجدت الاجابة Good idea!
والسؤال الان ما الرابط بين هذا النظام وبين هياكل البيانات ؟
ان الرابط العجيب هو الالكترون والذى يمكن ان يكون

- متغير (الذى اعتدت ان تتعامل معه فى كل برامجك)
- بيانات (متغير بدون اسم)

والسوال الكبير الان : ما الفائدة من وجود متغير بدون اسم طالما لم نستطيع الاشارة اليه للتعامل معه ؟

جـ : حقا ان المتغير بدون اسم – لكن يمكن الاشارة اليه والتعامل معه من خلال اسم الذرة – اسم المدار – اسم المدار الفرعى

س : عفوا – هل يمكن ان يشمل المدار الفرعى اكثر من الكترون ؟
جـ : بالتاكيد من الممكن ذلك

س : اذن ماذا لو كانت هذه الالكترونات عبارة عن بيانات اى متغيرات بدون اسماء كيف سوف نميز بينها طالما انها فى نفس المدار الفرعى ؟

جـ : ببساطة سوف يتم ذلك من خلال التنقل بين المتغيرات كما لو كنت تتنقل بين سجلات ملف بيانات او بين عناصر مصفوفة.

من خلال فهمك لهذا التنظيم تجد ان المتغيرات او البيانات اصبحت تحت السيطرة بمجرد تواجدها داخل مدارت فرعية تنتمى لمدارات اساسية
متواجدة ضمن ذرات – والسيطرة تعنى انك تتعامل مع متغيرات النظام بدون الحاجة لمعرفة اى شى مسبق عنها – فانت لست بحاجة لمعرفة
اسم المتغير – كما انك يمكنك ان تلعب بالمتغيرات وتنقلها من مكان لاخر (من ذرة لاخرى) وغير ذلك من الامور الكثيرة المثيرة التى سوف تندهش كثيرا عندما نتعرض لها.

نظام إدارة قاعدة البيانات التخيلى Virtual DBMS :-

هو نظام مبنى على النظام الكيميائى يهدف الى توفير نظام إدارة قاعدة بيانات تخيلى لان يعمل فى الذاكرة العشوائية RAM ويتيح لك هذا النظام عمل ملفات بيانات ذات مواصفات خاصة ويمكن التعامل مع هذه الملفات باجراء العمليات المختلفة مثل إضافة وتعديل السجلات بالاضافة الى عمليات البحث.

ان قاعدة البيانات هى افضل وسيلة لادارة البيانات ووجود شبيه لهذه الوسيلة للتعامل مع البيانات بصفة موقته فى RAM امر جيد له العديد من الاستخدامات مثل عمل نظم ملفات File System خاص بك يتم ادارته بسهولة من قبل قاعدة البيانات التخيلية مما يعطى سرعة عالية اثناء العمل.

مثال على ذلك تخيل برنامج مثل Microsoft Word الذى يحتاج الى قاعدة بيانات خاصة به لادارة المستند الذى تحرره اثناء عملك عليه لهذا فان هذا البرنامج ينشى مجموعة من ملفات البيانات على وحدة التخزين Hard Disk عندما يقوم بفتح ملف Doc وعند اغلاق الملف – يقوم بحذف هذه الملفات الموقتة – بدلا من ذلك فان قاعدة البيانات التخيلية توفر الحاجة الى Hard disk من اجل التخزين الموقت وتستخدم RAM بدلا منه.

ايضا ان وجود مفهوم قاعدة البيانات التخيلية يعطى دعم كبير لهياكل البيانات اثناء عملية التصميم.
والسوال الان : كيف يتم تمثيل قاعدة بيانات من خلال قاعدة البيانات التخيلية

How chemical system presents virtual DBMS?

Data File	Atom (2 sub shells)
Record	Electron of type DataBlock
Relations & Filters	Reaction
Database container	Vessel

For defining virtual data file details

```
Atom Telephone
    Shell K
        SubShell S
            Var Name_C_50
            Var Address_C_50
            Var Telephone_C_20
        SubShell P
```

نلاحظ اننا نحتاج مدارين فرعين فقط لتمثيل ملف البيانات
١ – مدار يشمل Details الخاصة بملف البيانات
حيث يتم تحديد اسم Coulum/Attribute/Field ونوعة Type وسعته Size
وستم الفصل بين الاسم والنوع والسعة من خلال العلامة بين الاقواس (_) اى Underscore
٢ – مدار يشمل Data اى السجلات الخاصة بملف البيانات
هو مدار يحتوى على الكترونات من نوع Data يقوم نظام ادارة قاعدة البيانات التخيلى بتنظيم التعامل
معها تبعا لمواصفات ملف البيانات التى يتم تحديدها.

فى المثال السابق نلاحظ ان لدينا ذرة تحمل الاسم Telephone وتشتمل على المدار الرئيسى K والذى بدوره يشمل المدار الفرعى S والمدار الفرعى P
تم تخصيص المدار S كى يحمل مواصفات ملف البيانات بحيث يشمل ثلاثة حقول هى Name, Address & Telephone اما المدار P فقد تم تخصيصة لكى يحمل سجلات ملف البيانات التخيلى.

اما بخصوص كيفية تحديد اسم ملف البيانات التخيلى – وكيفية اضافة البيانات والبحث عنها وتعديلها فسوف يتم التعرض لذلك لاحقا عند ادراك كيفية التعامل مع وحدة البيانات.

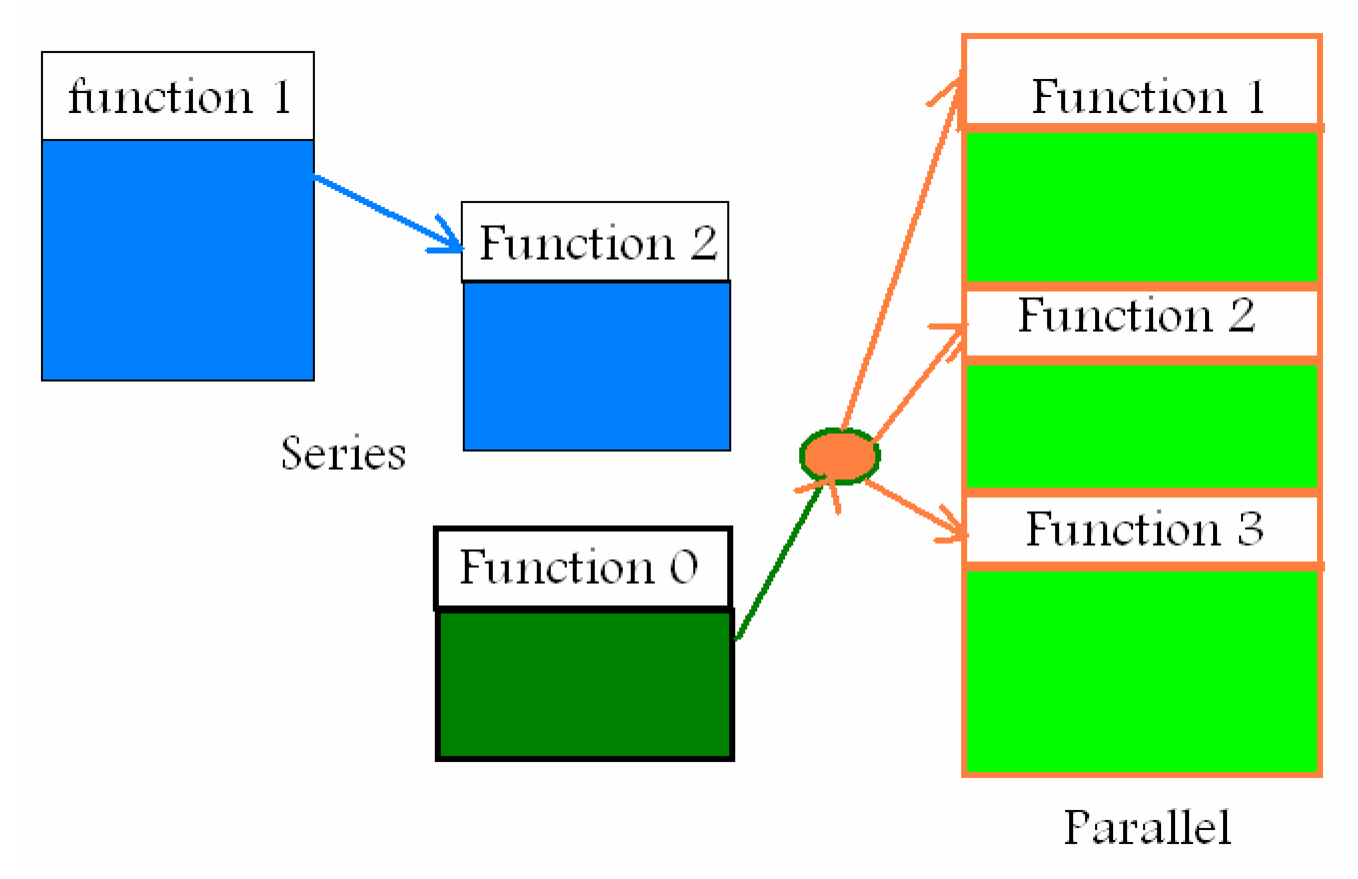
ملحوظة هامة

الذرة الواحدة يمكن ان تشمل ١٤ ملف بيانات تخيلى لان الذرة الواحدة تشمل ٧ مدارات رئيسية وكل مدار رئيسى يشمل ٤ مدارات

فرعية اى لدينا ٢٨ مدار فرعى فى الذرة – بينما نحن بحاجة الى مدارين فقط لعمل ملف بيانات تخيلى.

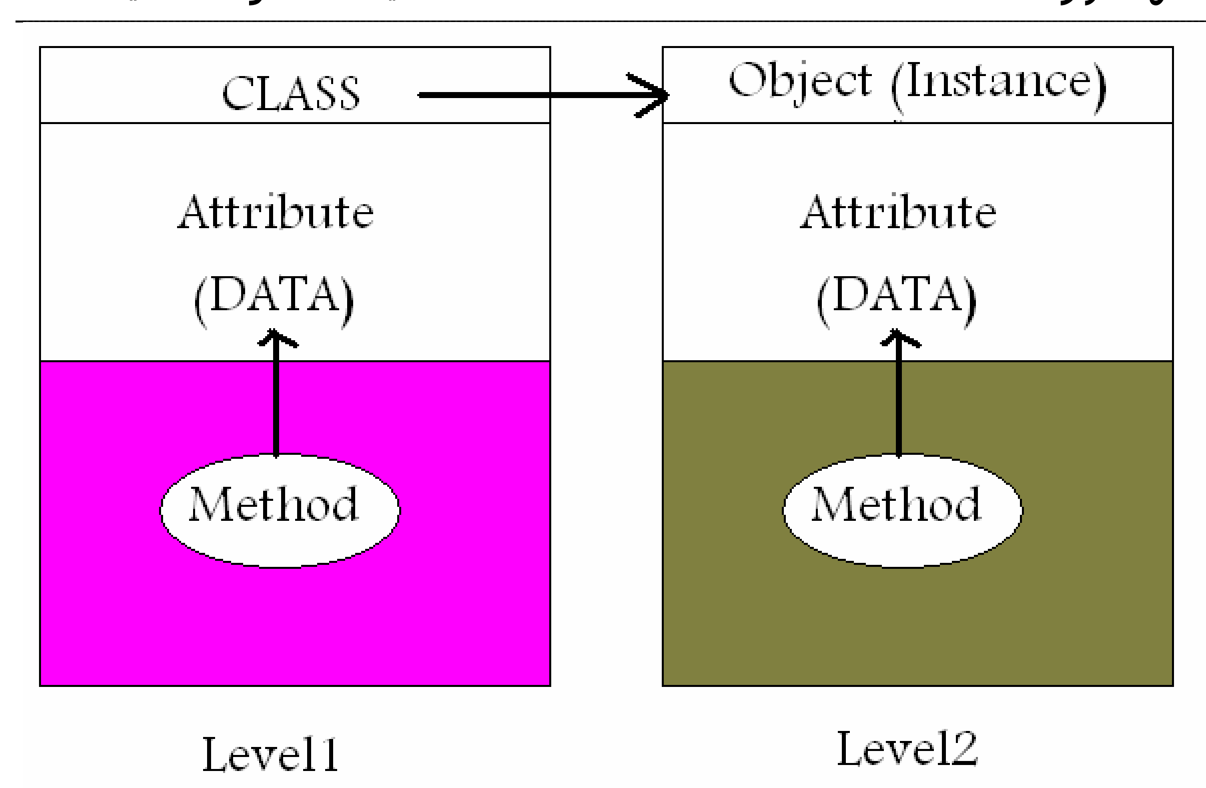
مفهوم وحدة التعليمات او الكود Code Unit Concept:-

قامت البرمجة الهيكلية على وضع الدالة فى الاعتبار كوحدة بناء اساسية يتم تكوين النظام من خلال مجموعة منها تعمل معا بصورة جيدة من التوافق سواء كانت دوال تعمل على التوالى (واحدة تنادى الاخرى) او على التوازى (دوال مجتمعة معا فى مكتبة Library وتعطى مفهوم متضامن واحد) وتصنيف التوالى والتوازى هنا من حيث النداء وليس التنفيذ .



شكل (٢٣) نداء الدوال فى البرمجة الهيكلية لبعضها البعض

تطور المفهوم مع برمجة الكائنات بحيث تتطور الدالة Function وتصبح طريقة Method وعندها يضاف ملامح جديدة للدالة التى تكون مخصصة للتعامل مع بيانات محددة يتم تحديدها على مستويين – المستوى الاول وهو عام من خلال تحديد سمات الفصيلة اى Class Attributes والمستوى الثانى عند تحديد قيم الكائن Object Properties.



شكل(٢٤) :- الـ Method تصمم بصفة عامة داخل Class وتعمل بصفة خاصة على بيانات الكائن

وعلى الرغم من تطور المفهوم فى برمجة الكائنات ليشمل مرونة فى التعامل مع البيانات الا ان ذلك التطور لم يعد كافيا الان فنحن بحاجة الا نظم تعمل باستمرار وفى نفس الوقت تستجيب لاحداث متغيرة – نحن الان نعمل فى بيئة تعتمد على الاحداث التى هى عبارة عن شروط يتم فحصها واذا تحققت يتم مناداة تعليمات او اكواد محددة – ان هذه التعليمات او الاكواد ماهى الا دوال Functions او طرق Method ولكننا نريد مستوى اعلى من التعريف فى نمط البرمجة ليقدم مفهوم Event الذى هو عبارة عن :-

CHECK ALWAYS (CONDITION), IF TRUE (CALL FUNCTION/METHOD)

ان إضافة مفهوم الـ Event الى نمط البرمجة امر ضرورى من الناحية النظرية من اجل رفع طبقة التوضيح Abstract الخاصة بالتصميم.

ان نمط البرمجة DoubleS من خلال وحدة التعليمات – يدعم مفهوم اخذ الاحداث فى الاعتبار ويتيح امكانية عمل احداث جديدة
وكتابة التعليمات الخاصة بها بصورة منظمة واضحة فى كل من مرحلتى التصميم والتطبيق.

ان وحدة التعليمات تقدم بديل لكل من Function و Method هذا البديل هو المقاومة Resistance – وحتى لا يختلط الامر فان
المقاومة Resistance ليست دالة Function عادية كما انها ليست طريقة Method
وايضا ليست حدث Event وانما هى عبارة عن مكون جديد مختلف يمكن تشكيله
حسب الحاجة بمرونة بحيث يكون دالة Function او طريقة Method او حدث Event

شكل(٢٥) مفهوم المقاومة فى وحدة التعليمات داخل نمط البرمجة DoubleS

تعريف المقاومة Resistance :-
" هى مجموعة من التعليمات التى يتم مناداتها تلقائيا و بصورة تكرارية اثناء وقت التشغيل "

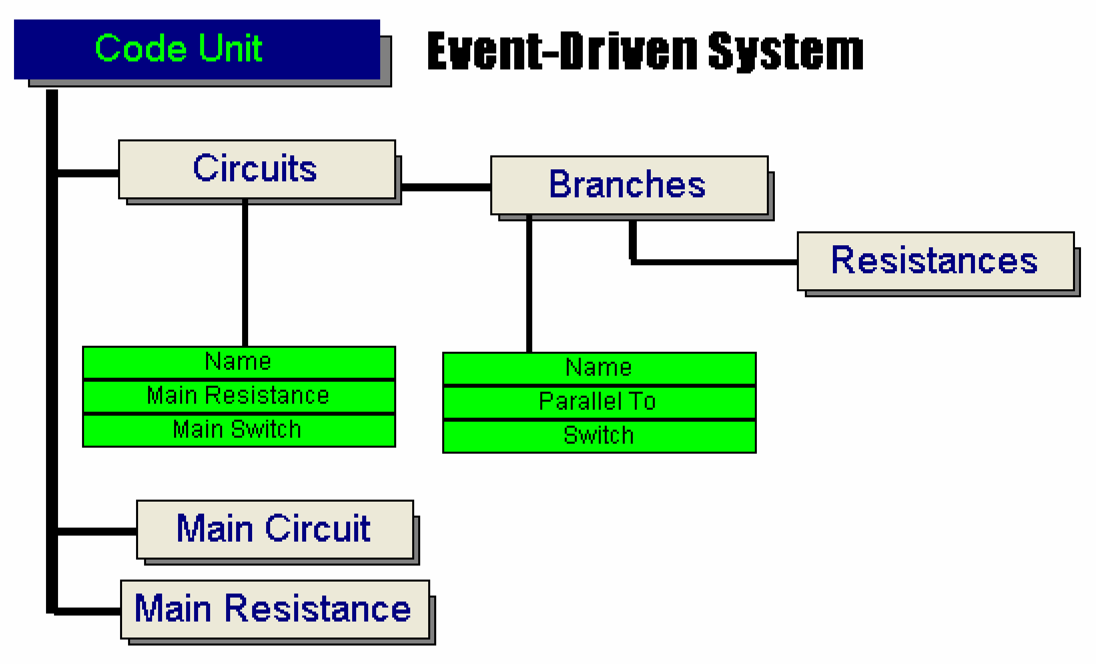
- المقاومة كدالة Resistance As Function :-
  - إذا تم مناداتها بحيث تعمل مرة واحدة وتم إلغاء او ايقاف صفتى التلقائية والتكرارية
- المقاومة كطريقة Resistance As Method :-
  - إذا تم إستخدامها من خلال محاكاة برمجة الكائنات كاحد المقاومات داخل فرع مستخدم لتعريف فصيلة Class
- المقاومة كحدث Resistance As Event :-
  - إذا تم اضافة فحص لشرط فى بدايتها Check for Condition

ملحوظة هامة

ان سهولة تشكيل المقاومة Resistance بحيث تكون دالة او طريقة او حدث امر فى غاية الاهمية فهو يدعم سمات التطبيقات والنظم المتطورة وفى نفس الوقت يسهل عملية محاكاة النظم البسيطة ولا يضيف اليها اى نوع من انواع التعقيد عن تمثيلها من خلال نمط البرمجة DoubleS.
هذا من ناحية – ومن ناحية اخرى فان وحدة البيانات Code Unit ليست بتلك البساطة – فهى لا تشمل فقط على المقاومات بل تقدم الفرع Branch وتقدم الدوائر Circuits وتقدم المفاتيح Switches وتقدم مفهوم Main Resistance المسئولة عن التحكم Control وسوف يتم التعرض لهذه المفاهيم ومعرفة كيفية الاستفادة منها للحصول على النتائج المطلوبة بكفاءة عالية.

محاكاة علم الدوائر الكهربية Electrical Circuit Simulation :-

علمنا ان المقاومة هى عبارة عن مجموعة من التعليمات التى يتم مناداتها تلقائيا بصورة تكرارية و هاتين الصفتين رغم فائدتهما الا انه ينبغى التحكم بهما لتحقيق النتائج المطلوبة ويتم ذلك من خلال صورة كاملة لنظام مرن نحصل عليها بمحاكاة الدوائر الكهربية



شكل(٢٦) – وحدة البيانات

الفرع Branch :-
" مجموعة من المقاومات معا تحت مسمى واحد – يشمل الفرع على مفتاح Switch كما يتم تحديد مكانه بتحديد عنصر يكون هذا الفرع موازيا له – وهذا العنصر اما ان يكون Branch اخر او مقاومة Resistance "

الدائرة Circuit :-
" مجموعة من الفروع معا تحت مسمى واحد – وتشمل الدائرة على مفتاح رئيسى ومقاومة رئيسية "

المفتاح Switch :-
" متغير منطقى Logic اذا كان متحقق True يتم تنفيذ محتويات العنصر الذى يبدا بهذا المفتاح سواء كان فرع او دائرة "

الدالة الرئيسية Main Resistance داخل دائرة :-
" يتم مناداتها باستمرار اثناء عمل الدئرة قبل وبعد اى فرع او مقاومة وفى بداية ونهاية عمل الدائرة "

اى هناك ٦ احتمالات :-

- ١ – بداية عمل الدائرة
- ٢ – بداية فرع
- ٣ – بداية مقاومة
- ٤ – نهاية مقاومة
- ٥ – نهاية فرع
- ٦ – نهاية عمل الدائرة

الدالة الرئيسية Main Resistance داخل وحدة البيانات :-
" يتم مناداتها مرة واحدة فقط عند بداية عمل الخادم "

الدائرة الرئيسية Main Circuit :-
" تحدد نقطة البداية فهى الدائرة التى يبدا الخادم العمل من عندها ثم ينتقل الى الدوائر الاخرى بالترتيب "

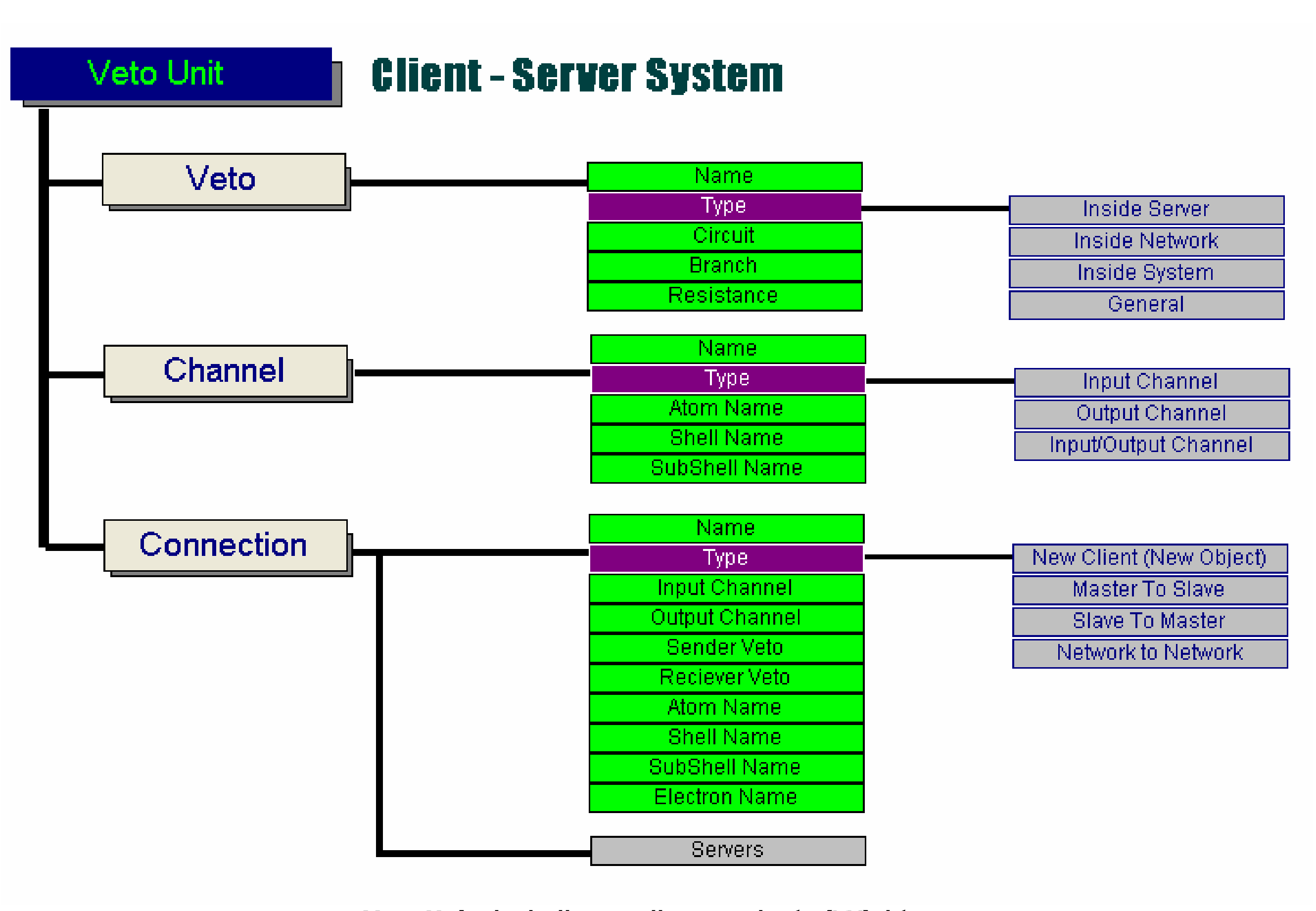
ملحوظة هامة

يمكن الرجوع للباب الاول من هذا الكتاب (نموذج سير العمليات) للحصول على معلومات كافية عن البنية الاساسية خلف وحدة التعليمات او الاكواد Code Unit

مفهوم وحدة النقض\ التراسل Veto Unit Concept:-

نحن فى عالم تطبيقات الزبون-الخادم Client-Server و التطبيقات الموزعة Distributed Application وتطبيقات الحساب المتوازى Grid Computing وهذه التطبيقات ترتكز جميعها على بنية اساسية واحدة هى التراسل بين المكونات المنفصلة عن بعضها البعض.

جاء نمط البرمجة Agent Oriented اخذا هذا المفهوم فى الاعتبار ولكنه اتى بصورة عامة غير تفصيلية وليست متعمقة مثلما فعل نمط البرمجة الخادم الممتاز DoubleS الذى جاء بصورة تفصيلية لميكانيكية التراسل بين المكونات الموزعة والمفصولة عن بعضها البعض – تم ذلك من خلال وحدة خاصة تسمى وحدة النقض او التراسل Veto Unit وهذه الوحدة تتيح وضع نموذج لنقل كل من البيانات Data ورسائل النقض Veto بين المكونات المختلفة.



شكل(٢٧) مكونات وحدة النقض والتراسل Veto Unit

تشتمل وحدة النقض على ٣ مكونات اساسية هى :-

- النقض Veto
- القناة Channel
- الاتصال Connection

النقض Veto :-

"هو تعريف لرسالة Message ممكن ان يستقبلها الخادم لاداء مهمة معينة والفرق بين النقض Veto والرسالة Message هو ان النقض ممكن ان يتم رفضه ولا يستجاب له."

يتم تحديد نوع للنقض Type ليبين مدى ظهور امكانية استقبال هذه النقض – اما داخل الخادم او الشبكة او النظام او عام.
ان النقض مجرد وصف للعالم الخارجى – لهذا يتم تحديد مقاومة Resistance يتم استدعائها بمجرد استقبال هذا النقض من العالم الخارجى.

القناة Channel :-

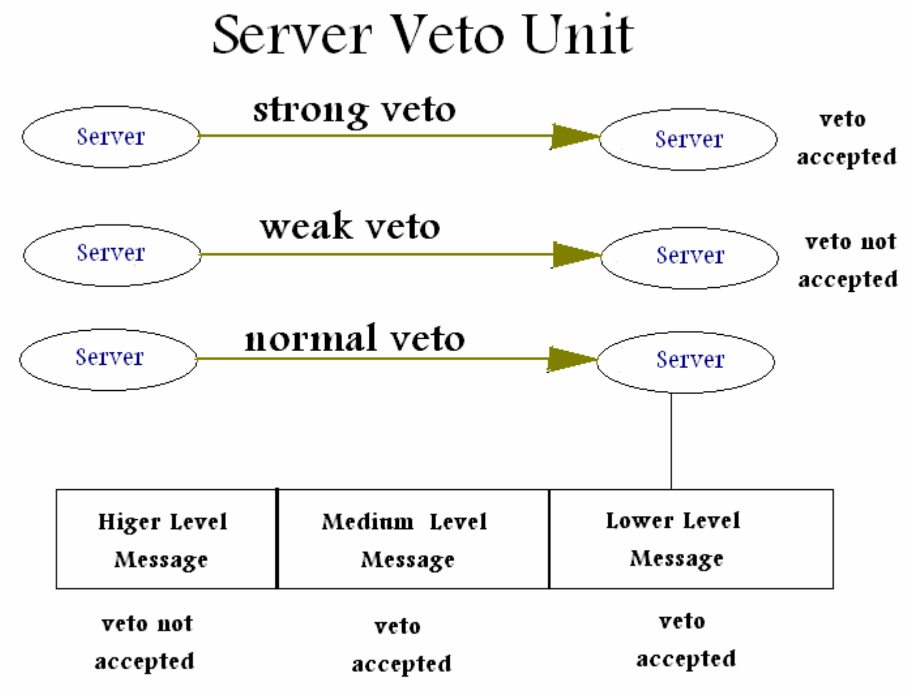
" هى عنوان يشير الى مكان تخزين اى بيانات او نقض يرسله او يستقبله الخادم "
القناة هى مجرد اسم ظاهرى داخل وحدة النقض يشير على مدار فرعى داخل احد الذرات داخل الخادم وهنا يتضع مفهوم المدار الفرعى الذى يشمل عدد من الالكترونات

الى هنا فى هذه الحالة قد تكون بيانات Data او نقض Veto قام الخادم بارساله او استفباله القناة ايضا لها انواع – قناة ادخال Input Channel لما يستقبله الخادم وقناة اخراج Output channel لما يرسله الخادم.

الاتصال Connection :-
" مجموعة من المعلومات التى تحقق الاتصال الناجح بين الخادم وخادم اخر او مجموعة من الخوادم الاخرى "
يتم اسناد الكترون (متغير) ليحمل مسئولية الاشارة الى الاتصال Connection كما يتم تحديد قناة الادخال والاخراج بالاضافة
الى النقض المرسل(يتم استدعائه داخليا داخل الخادم عندما يبدا عملية الارسال) والنقض المستقبل (يتم استدعائه داخليا
داخل الخادم عندما تبدا عملية الاستقبال).

محاكاة مفهوم التفاعل الانسانى Human Interaction Simulation :-

يظهر ذلك فى وحدة النقض والتراسل Veto Unit عندما يتم ارسال نقض Veto من خادم لاخر حيث تكون عملية الاستجابة او الرفض مشابهة لمفهوم التفاعل الانسانى – فعندما يقدم احد طلب لشخص اخر قد يتم القبول او الرفض كما ان عملية القبول او الرفض تتعلق باسباب مختلفة اهمها العلاقة بين الشخص الذى يطلب والشخص الذى يفترض ان يقدم الطلب فمثلا يفترض ان تستجيب لطلبات رئيس العمل لانها ليست مجرد طلبات – بل تعد اوامر او تعليمات – ام طلب زميلك فى العمل فقد يحتمل القبول او الرفض اما طلب من شخص مجهول لاتثق به فمن الجائز جدا ان يتم رفضه اذا لم تكن الخدمة عامة وبسيطة.



شكل(٢٨) :- عملية الاستجابة والرفض داخل وحدة النقض والتراسل Veto Unit

ونلاحظ من الشكل ان النقض Veto يمكن تسميته تبعا لاحتمالية قبوله او رفضه – فمثلا النقض القوى Strong Veto دائما يتم الاستجابة الايجابية له بالقبول – ام النقض الضعيف Weak Veto فينال الاستجابة السلبية له بالرفض اما النقض العادى Normal Veto فيحتمل ان يتم قبوله او رفضه حسب مكنون النقض

ملحوظة هامة

حتى تتم عملية التميز بين الخوادم وبعضها البعض تظهر لنا خاصية القيمة المميزة Eigen Value والتى يتم اعطائها للخادم حتى يتم التفرقة بين الخوادم وبعضها البعض والجدير بالذكر ان القيمة المميزة هنا ليست رقم تعريف ID لكل خادم على حدة – بل يمكن ان يتشارك مجموعة من الخوادم فى Eigen Value.

مفهوم جمل المقاومات Resistance Statements :-

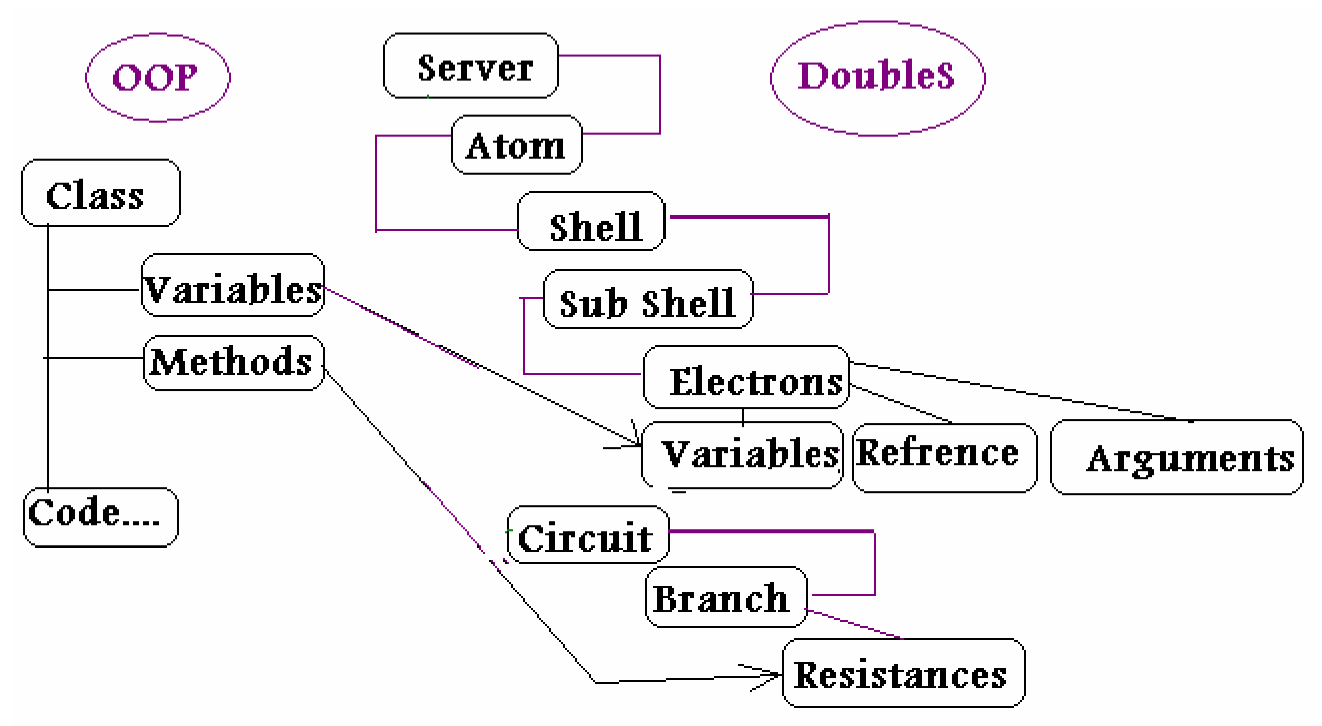
نظرا لان تركيب الخادم يعد معقدا بشكل ما من الاشكال – لذا فلابد من كسب الفائدة المرجوة من وراء هذا التعقيد يتم ذلك من خلال مايسمى جمل المقاومات Reistance Statements ويقصد بها مجموعة كبيرة من الاوامر التى يتم اضافتها الى لغة البرمجة حتى تتعامل مع مكونات الخادم المختلفة – وسميت جمل المقاومات لان هذه الجمل البرمجية يتم كتابتها داخل التعليمات او الاكواد الخاصة بالمقاومات التى يمكن ان تمثل دوال او طرق او احداث حسب طبيعة الاستخدام.

محاكاة برمجة الكائنات OOP Simulation :-

نحن الان نتواجد فى عالم من البرمجيات يرتكز على برمجة الكائنات لهذا توجد الملايين من البرامج والتطبيقات التى تم تصميمها باستخدام برمجة الكائنات واعادة تصميم هذه البرامج لكى تكون بنيتها الاساسية الخادم الممتاز امر ليس سهلا ابدا اذا كنا سوف نعيد اختراع العجلة ونعيد تصميم كل الانظمة من الصفر – لهذا يقدم لنا نمط برمجة الكائنات ما يسمى بمحاكاة برمجة الكائنات وهى حل مثال لنقل النظم التى تم تصميمها مرتكزة على برمجة الكائنات الى عالم الخادم الممتاز DoubleS بدون اى عقبات او بدون الحاجة الى تغيير تصميم النظام وبعد عملية النقل واتمامها بنجاح – تظهر فرصة امكانية جلب المكاسب من وراء نقل النظام الى الخادم الممتاز من حيث سهولة تحويل النظام الى Client-Server او Application Distributed او Grid Computing او التحكم فى خصائص نظام ادارة الاحداث Event-Driven System بسهولة – او ادارة هياكل بيانات معقدة Complex Data Structure بمرونة وغيرها الكثير من ملامح نمط البرمجة الخادم الممتاز

ان محاكاة برمجة الكائنات ترتكز على ان يتم معاملة الفصائل او الفئات Classes معاملة الكائنات Objects بحيث يتم انشائها فى وقت التشغيل Runtime ويظهر مع ذلك امكانية تعديل سمات الفصيلة Class اثناء عمل البرنامج – وقد يصرخ البعض مستفسرا عن سبب الحاجة الى ذلك فتكون الاجابة ان النظام فى الخادم الممتاز يتعامل مع العالم الخارج ويتراسل معه وهذا العالم الخارجى هو عالم متغير – ويفترض بالخادم ان يكون متغيرا ليستجب لتغيرات هذا العالم المحيط وبالتالى فان الحصول على فصائل

ديناميكية امر فى غاية الاهمية يضيف الكثير من المرونة الى برمجة الكائنات لتصبح بذلك برمجة الكائنات الموجه ذات الفصائل الديناميكية.



شكل(٢٩) :- محاكاة برمجة الكائنات داخل نمط برمجة الخادم الممتاز DoubleS

انظر شكل (٢٩) والذى يوضح اننا بحاجة فقط الى مدار فرعى Sub Shell واحد بالاضافة الى Branch واحد لتمثيل فصيلة ونحن نعلم ان الالكترونات داخل المدار الفرعى ديناميكية – يمكن اضافة الكترونات جديدة او حذفها مما يعنى امكانية عمل فصائل ذات خصائص ديناميكية.

وحيث ان الذرة من الممكن ان تشمل ٢٨ مدار فرعى (٧ مدارات اساسية – كل مدار اساسى يشمل ٤ مدارات فرعية) هذا يعنى ان ذرة واحدة + دائرة تشمل ٢٨ فرع من الممكن ان تمثل ٢٨ فصيلة.

هذا يعنى ان الخادم الواحد من الممكن ان يمثل المئات بل الالاف من الفصائل وهذا يعطى كبسلة اكثر More Encapsulation كما ان الفصائل من الممكن ان يتم انشائها لتتشارك فى القروع Branches او المدارات الفرعية مما يعنى سهولة انشاء الفصائل ذات الصفات المتشابهة بدون الحاجة الى الوراثة Easy to Create Simi Similar Classes وفى الواقع فان ذلك يقلل الحاجة الى الوراثة بنسبة كبيرة قد تصل الى ٥٠ % على مستوى النظام ككل Reduce Inheitance by 50%

تفاصيل إستخدام وحدة البيانات :-

لاعلان إلكترون جديد يتم ذلك بنسبة الالكترون الى ذرة محددة ثم مدار داخل هذه الذرة ثم مدار فرعى داخل المدار المحدد يلى ذلك نوع الالكترون ثم اسمه كالتالى

```
Atom <Atom_name>
          Shell <Shell_name>
              SubShell <Subshell_name>
                    Electron_Type  <Electron_name>
```

مثال على ذلك :-

```
Atom Customer
          Shell K
               SubShell S
                      Var Cust_Code
                           Var Cust_Name
                           Var Cust_Company
                           Var Cust_Telephone
```

حيث قمنا بعمل ذرة بالاسم Customers وتم تحديد المدار k ثم المدار الفرعى S لكى يحمل المتغيرات Cust_Code و Cust_Name و Cust_Company و Cust_Telephone بحيث ان هذه المتغيرات هى عبارة عن الكترونات من النوع (متغير) Electrons of type : (Variable) كما يتضح من المثال لهذا سبقت بالنوع Var

س :- هل يعنى ذلك انه لعمل متغيرات لابد من تحديد ذرة ومدار ومدار فرعى ؟
جـ :- نعم ويجب ان تضع فى الاعتبار ان هذه المتغيرات سوف تكون عضو دائم داخل هيكل بيانات الخادم (مالم يتم حذفها) لذلك
يجب ان تضعها بعناية فى الحسبان اثناء وقت التصميم – كما ان هذه المتغيرات يمكن الوصول اليها من قبل اى مقاومة Reistance
داخل الخادم – بمعنى ان هذه المتغيرات تكون متاحة لبقية اجزاء الخادم.

والسوال الان : كيف سوف تتم عملية الوصول لهذه المتغيرات ؟
جـ : ان ذلك يتم بمفهوم مشابهه لتقنية الزبون-الخادم حيث يتم تحميل نسخة من المتغيرات للتعامل معها وبعد ذلك تحدد هل سوف تكون هناك عملية تحديث Update للمتغيرات الاصلية ام لا.

- Resistance code :

```
Select Address Customer:K:S
Load active subshell from memory
Cust_code = 1
Cust_name = "Mahmoud Fayed"
Cust_company = "Microsoft"
Cust_telephone = "000"
Upload active subshell to memory
```

فى البداية يتم تحديد العنوان من خلال الامر Select Address ويتكون العنوان من اسم الذرة:اسم المدار:اسم المدار الفرعى يتم استخدام الجملة Load Active SubShell From Memory حتى يتم عمل نسخة من المتغيرات الموجودة فى المدار بنفس الاسم فى الذاكرة Ram حتى نتعامل معها باسمائها الحقيقية ولكن مع صورة غير اصلية من المتغيرات (المتغيرات الاساسية لا تتاثر بالعمليات التى نجريها)
يقال على هذه العملية we converted the public variables to local variables with the same name اى قمنا بتحويل المتغيرات العامة الى اخرى محلية بنفس الاسم – من ناحية مدى ظهور المتغيرات Scope
لعمل تحديث للمتغيرات الاصلية نستخدم الامر Upload Active SubShell to Memory

لقد عرفنا الالكترون من النوع : (متغير) – ماذا عن الانواع الاخرى للكترون ؟
جـ : يوجد النوع موشر او دليل Refrence والذى يشير الى عنوان بيانات او دالة ما.

اما النوع الثالث فهو وحدة البيانات DATABLOCK والذى يمكن ان نعتبره متغير بدون اسم مثال على ذلك

```
ATOM Mydata
          SHELL K
             SUBSHELL S
                    DATABLOCK ANYNAME
                    DATABLOCK ANYNAME
                    DATABLOCK ANYNAME
                    DATABLOCK ANYNAME
                    DATABLOCK ANYNAME
```

والمثال التالى يوضح كيفية التعامل مع مثل هذه الانواع من الالكترونات

- Resistance code :

```
select address Mydata:K:S
     Goto First Electron
     DO while Get_Electron_Num < Get_Electrons_Count
                    Load active subshell from memory
                    ? GET_ELECTRON_VALUE
                    Goto Next Electron
     Enddo
```

حيث يتم تحديد العنوان ومن ثم التنقل بين الالكترونات بصورة مبسطة.

سوف نناقش الان قاعدة البيانات التخيلية Virtual DBMS

Data File	Atom (2 sub shells)
Record	Electron of type DataBlock
Relations & Filters	Reaction
Database container	Vessel

نلاحظ ان ملف البيانات يمثله مدارين فرعيين داخل الذرة والسجل عبارة عن الكترون
مثال لتعريف ملف بيانات تخيلى

```
Atom Telephone
        Shell K
                SubShell S
                        Var Name_C_50
                        Var Address_C_50
                        Var Telephone_C_20
                SubShell P
```

ولانشاء ملف البيانات

- Resistance code :

```
NEW VIRTUAL DATA FILE mydata1 DETAILS telephone:k:s DATA telephone:k:p
```

ولفتح الملف

- Resistance code :

```
OPEN VIRTUAL DATA FILE mydata1
```

ولاضافة سجل جديد

- Resistance code :

```
ADD NEW RECORD
Load record
        Name = "Mahmoud Fayed"
        Address = "Egypt"
        Telephone = "000"
Upload record
```

تذكر جيدا ان كل ذلك يتم فى الذاكرة العشوائية RAM وليس على اقراص وحدات التخزين

ملحوظة هامة

ينبعى التفرقة بين الجمل التى تكتب داخل مقاومات Resistance Code
والجمل التى لا تكتب بداخلها وانما تستخدم فى تعريف مواصفات الخادم.

فيما يلى الجمل التى يمكن ان تكتب داخل المقاومات لكى تستعمل مع وحدة البيانات Data Unit

DoubleS Resistance Statements for Data Unit:

- SELECT ATOM <X>
- SELECT SHELL <X>
- SELECT SUBSHELL <X>
- SELECT ADDRESS <X>:<X2>:<X3>
- LOAD ACTIVE SUBSHELL FROM MEMORY
- UPLOAD ACTIVE SUBSHELL TO MEMORY
- GET_ACTIVE_ATOM
- GET_ACTIVE_SHELL
- GET_ACTIVE_SUBSHELL
- GET_ATOMS_COUNT
- GET_SHELLS_COUNT
- GET_SUBSHELLS_COUNT
- GET_ELECTRONS_COUNT
- GET_VESSELS_COUNT
- GET_REACTIONS_COUNT
- GET_CIRCUITS_COUNT
- GET_BRANCHES_COUNT
- GET_RESISTANCES_COUNT
- GET_VETOS_COUNT
- GET_CHANNELS_COUNT
- GET_CONNECTIONS_COUNT
- GET_ACTIVE_ELECTRON_ID
- GET_ACTIVE_ELECTRON_NUM
- GET_ACTIVE_ELECTRON_NAME
- GET_ACTIVE_ELECTRON_VALUE
- GET_ACTIVE_ELECTRONS_COUNT
- GOTO FIRST ELECTRON
- GOTO LAST ELECTRON
- GOTO NEXT ELECTRON
- GOTO PREV ELECTRON
- DELETE ACTIVE ELECTRON
- DELETE ALL ACTIVE ELECTRONS
- COPY ELECTRONS TO <X21>:<X22>:<X23>
- COPY ADDRESS <X>:<X2>:<X3> ELECTRONS TO <X21>:<X22>:<X23>
- MOVE ELECTRONS TO <X21>:<X22>:<X23>
- MOVE ADDRESS <X>:<X2>:<X3> ELECTRONS TO <X21>:<X22>:<X23>
- ADD MARK <X> TO ADDRESS <X2>:<X3>:<X4>
- CREATE ELECTRONS LIST <X>
- OPEN ELECTRONS LIST <X>
- CLOSE ELECTRONS LIST
- ADD ELECTRON TO LIST
- DELETE ELECTRON FROM LIST
- SET DOMAIN <electronlist>
- CLOSE DOMAIN
- NEW VIRTUAL DATA FILE <X1> DETAILS <X2>:<X3>:<X4> DATA <X5>:<X6>:<X7>
  OPEN VIRTUAL
  DATA FILE <X1>
- CLOSE VIRTUAL DATA FILE

- SELECT AREA <X1>
- LOAD RECORD
- UPLOAD RECORD
- ADD NEW RECORD
- DELETE THIS RECORD
- GOTO FIRST RECORD
- GOTO LAST RECORD
- GOTO NEXT RECORD
- GOTO PREV RECORD
- GET_RECORD_NUMBER
- GET_RECORDS_COUNT
- ACTIVE REACTION <X1>
- UNACTIVE REACTION <X1>
- SEARCH ABOUT <CONDITION>
- SEARCH OTHER
- THERE_ARE_RESULT
- GET_ACTIVE_AREA_NUM
- GET_ACTIVE_VIRTUAL_DATA_FILE_NAME

ان الجمل التى تستخدم مع وحدة البيانات تحقق الفائدة المرجوة من وراء وحدة البيانات وهذه الجمل قد تكون قابلة للتغيير من لغة برمجة الى اخرى تبعا لمصممى لغة البرمجة – ولكن هذه الجمل التى تم عرضها قد تم تطبيقها بالفعل مع اللغات من العائلة xBase مثل كليبر CA-Clipper و اكس بيس بلس بلس ++ xBase و اكس هاربور xHarbour و تم اختبارها ايضا مع xHarbour/MiniGUI

ان اغلب هذه الجمل مفهومة المعنى ويمكن الاستدلال على معناها واستخدامها بسهولة .

تفاصيل استخدام وحدة التعليمات او الاكواد Code Unit:-

بالنسبة لتعريف دائرة داخل الخادم – يتم ببساطة كالمثال التالى

```
Circuit      Main
          MainResistance CONTROL
          MainSwitch    On
          Branch    Main
            ParallelTo 0
              Switch    On
                          Resistance R1
                          Resistance R2
                          Resistance R3
                          Resistance R4
                          Resistance R5
```

فى هذا المثال تم تعريف دائرة Circuit بالاسم Main – بحيث تكون المقاومة الرئيسية هى CONTROL وتم ضبط المفتاح الرئيسى لكى تعمل الدائرة MainSwitch On

تم تعريف فرع داخل هذه الدائرة وتم تسميته بالاسم Main ايضا وتم ضبته ليكون موازيا للفرع الرئيسى وتم ضبط مفتاح هذا الفرع لكى يعمل – وتم اضافة خمسة مقاومات الى هذا الفرع هما R1,R2,R3,R3 & R5

وبعد ذلك يتم اسناد دوال – لكى تمثل المقاومات كالتالى

```
Resistance R1() Address Code Unit : Circuits\Main\Main\R1
Resistance R2() Address Code Unit : Circuits\Main\Main\R2
Resistance R3() Address Code Unit : Circuits\Main\Main\R3
Resistance R4() Address Code Unit : Circuits\Main\Main\R4
Resistance R5() Address Code Unit : Circuits\Main\Main\R5
```

ومن ثم يمكن كتابة التعليمات او الاكواد الخاصة بالمقاومات كالتالى

```
Function R1()
            ….code
 Return
```

Or, we can write

```
Resistance R1() code
      …..code
Return
```

ويجب ان لا ننسى تعريف الدالة التى تمثل المقاومة الرئيسية

```
Resistance CONTROL() Address Code Unit : CONTROL
```

ونكتب التعليمات الخاصة بها كالتالى

```
Resistance Control() Code
                        ...code
Return
```

ولتشغيل الخادم حتى تعمل الدائرة وباستمرار حتى تجد مايوقفها

```
Server FireOn
```

ولايقاف الخادم عن العمل نستخدم الجملة Server Shutdown

وبالمثل يوجد العديد من الجمل التى تستخدم للتعامل مع وحدة التعليمات او الاكواد – ويمكن كتابة هذه الجمل مباشرة داخل التعليمات او الاكواد الخاصة بالمقاومات

DoubleS Resistance Statements for Code Unit:

- SERVER FIREON
- SERVER SHUTDOWN
- WITH CIRCUIT <X1>
- SET MAIN RESISTANCE = <X1>
- SET MAIN SWITCH ON
- SET MAIN SWITCH OFF
- RESTART
- SET DIRECTION DOWN
- SET DIRECTION UP
- SET DIRECTION DOWN UP
- SET DIRECTION UP DOWN
- END WITH CIRCUIT
- WITH BRANCH <X1>
- SET SWITCH ON
- SET SWITCH OFF
- SET PARALLEL = <X1>
- FIREON ME
- GET_BRANCH_SWITCH_STATUS
- GET_PARALLEL
- END WITH BRANCH
- GET_ACTIVE_CIRCUIT
- GET_ACTIVE_BRANCH
- GET_ACTIVE_RESISTANCE
- ADD RESISTANCE <X1>
- DO SYSTEM EVENTS
- SLEEPTIME <X1>

تفاصيل إستخدام وحدة النقض Veto Unit :-

لتعريف نقض Veto داخل الخادم يتم استخدام الكلمة Veto يليها اسم النقض الذى نريده ويمكن ان ستقبله الخادم من خادم اخر ثم بعد ذلك يحدد نوع النقض بعد الكلمة Type ثم يحدد عنوان المقاومة التى سوف يتم استدعائها داخل الخادم اذا تم استقبال هذا النقض – انظر المثال التالى الذى يعرف نقض Veto يحمل الاسم ShowData وهو من النوع General اى عام ويعنى ذلك ان هذا النقض يمكن استقباله من اى خادم – وتم تحديد المقاومة R1 الموجودة فى الفرع B1 داخل الدائرة C1 لكى تكون هى المسئولة عن الاستجابة لهذا النقض.

```
Veto SHOWDATA
        Type General
        Circuit    C1
        Branch     B1
        Resistance R1
```

ولتعريف قناة Channel نستخدم الكلمة Channel يليها اسم القناة ثم بعد ذلك نحدد نوع القناة بعد الكلمة Type (اما قناة ادخال Input او قناة إخراج Output) ثم بعد ذلك نحدد المدار الفرعى الذى سوف يحمل البيانات التى يتم استقبالها فى حالة قناة ادخال او البيانات التى سوف يتم ارسالها فى حالة قناة إخراج – ويتم تسجيل البيانات داخل المدار الفرعى على هيئة إلكترونات من النوع وحدةبيانات DataBlock.

انظر المثال التالى الذى ينشى قناة اسمها MYINPUTCHANNEL من النوع (قناة إدخال) وتم إختيار المدار الفرعى S داخل المدار الرئيسى K الموجود بالذرة CHANNEL لكى يحمل البيانات التى يتم استقبالها.

```
Channel MYINPUTCHANNEL
        Type     Input Channel
        Atom     CHANNEL
        Shell    K
        SubShell S
```

ولتعريف إتصال يتم إختيار اسم له بعد الكلمة Connection ثم بعد ذلك نحدد نوع الاتصال بعد الكلمة Type ونحدد قناة الادخال و قناة الاخراج الخاصة بالاتصال بعد كلمتى InputChannel و OutputChannel ونحدد إلكترون معين من النوع (متغير Variable) لكى يستخدم فى الوصول الى هذا الاتصال الذى يتم تعريفه ثم فى النهاية تحدد الخوادم Servers التى نود الاتصال معها.

انظر المثال التالى والذى يعرف اتصال اسمه MyClientConnection من النوع New Client (New Object) وتم اختيار الالكترون Myclient من النوع (متغير) الموجود داخل المدار الفرعى S المتواجد فى المدار الرئيسى K داخل الذرة Clients للوصول للاتصال حتى يمكن استخدامه وتم اختيار الخادم MYSERVER حتى يتم الاتصال به من خلال هذا الاتصال.

```
Connection MyClientConnection
        Type          New Client (New Object )
        OutputChannel mychannel
        Atom          Clients
        Shell         K
        SubShell      S
        Electron      myclient
        Server        MYSERVER
```

اما بخصوص إستخدام الاتصال داخل المقاومات فيكون كالتالى
حيث نستخدم الامر SELECT CLIENT CONNECTION يليه عنوان الالكترون الذى يستخدم للتعامل مع الاتصال – ثم بعد ذلك يتم فتح الاتصال بالامر CON_CONNECT ثم يتم إرسال بيانات Data او نقض Veto الى الخادم الذى تم الاتصال به – ثم بعد ذلك يتم إنهاء الاتصال بالامر CON_DISCONNECT .

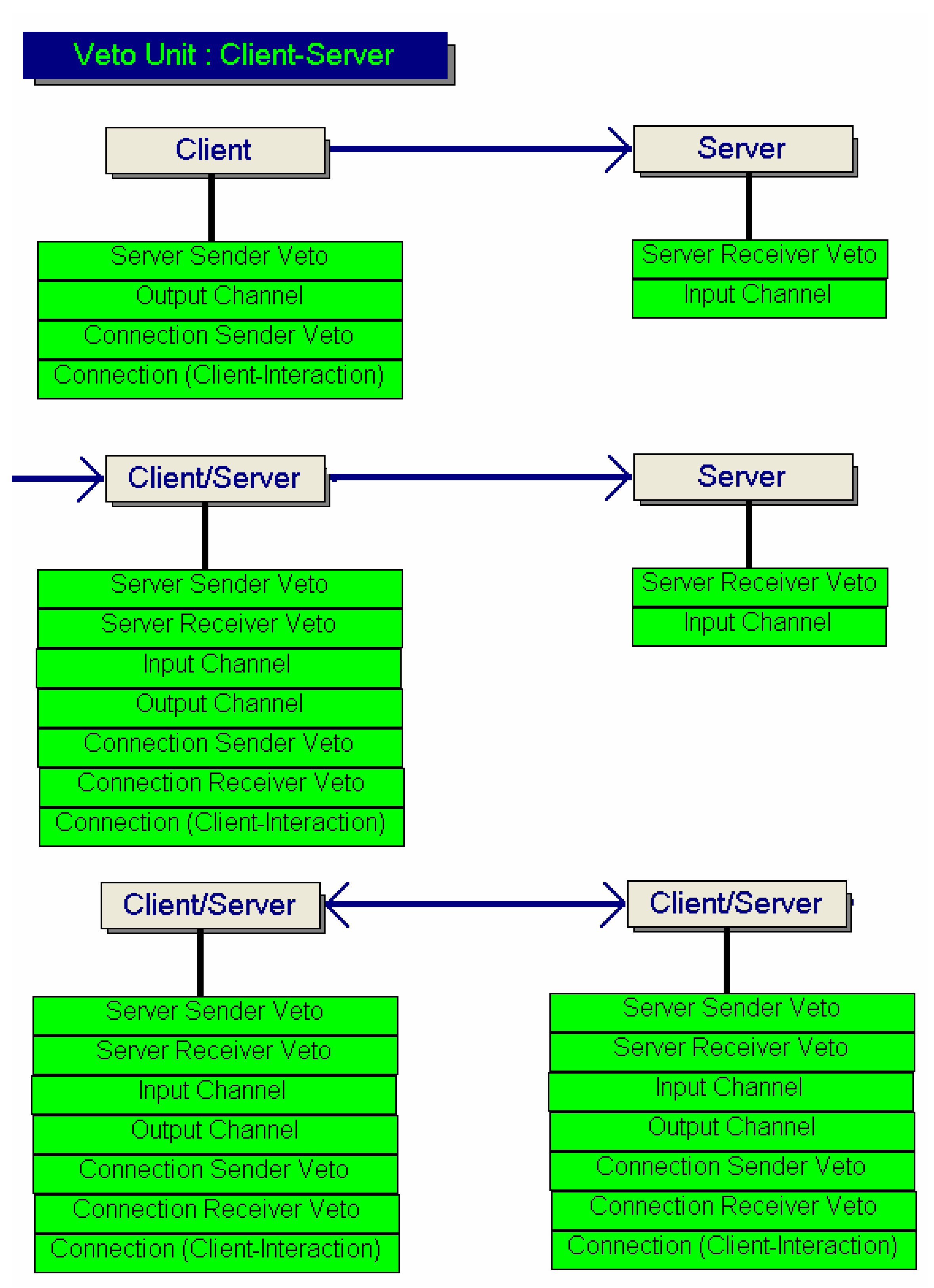
SELECT CLIENT CONNECTION CLIENTS:K:S:MYCLIENT
CON_CONNECT
CON_SENDDATA "THIS MESSAGE SENDED FROM CLIENT TO SERVER"
CON_SENDVETO SHOWDATA
CON_SENDDATA "ANOTHER MESSAGE FROM CLIENT TO SEREVER"
CON_SENDVETO SHOWDATA
CON_DISCONNECT

وتحتوى وحدة النقض على العديد من الجمل التى يمكن استخدامها فى كتابة التعليمات الخاصة بالمقاومات.

DoubleS Resistance Statements for VETO Unit:

- SELECT INTERACTION CONNECTION <connection name>
- SELECT CLIENT CONNECTION <atom>:<shell>:<subshell>:<electron_name>
- CON_CONNECT
- CON_BEGIN_TRANSACTION
- CON_SENDVETO <veto name>
- CON_SENDDATA <datablock>
- CON_END_TRANSACTION
- CON_SELECT_INPUT_CHANNEL
- CON_SELECT_OUTPUT_CHANNEL
- CON_CHANNEL_PUTDATA FROM <atom>:<shell>:<subshell>
- CON_CHANNEL_GETDATA FOR <atom>:<shell>:<subshell>
- CON_DISCONNECT
- SELECT INPUT CHANNEL <Channel_Name>
- OPEN CHANNEL <CHANNEL_NAME>
- CLEAR CHANNEL
- CLOSE CHANNEL
- GET_SENDER_SERVER_NAME
- GET_SENDER_SERVER_TYPE
- GET_SENDER_SERVER_EIGENVALUE
- GET_VETO_DECISION
- ACCEPT CONNECTION
- REFUSE CONNECTION
- CONNECTION_ACCEPTED
- REQUEST_TYPE_CONNECTION
- REQUEST_TYPE_SENDDATA
- REQUEST_TYPE_SENDVETO
- REQUEST_VETO_NAME
- STOP_SENDING
- GET_VETO_SYSTEM_LEVEL
- SET VETO SYSTEM LEVEL <X1>
- CHECK_VETO_SERVICES
- SET VETO SYSTEM PATH <X1>
- SET SERVER IP <X1>
- GET_SERVER_IP
- SET SERVER PROTOCOL
- GET_SERVER_PROTOCOL

والان انظر شكل (٣٠) الذى يبين مايلزم تعريفه داخل وحدة النقض Veto Unit حتى يعمل الخادم كزبون Client او كخادم Server.

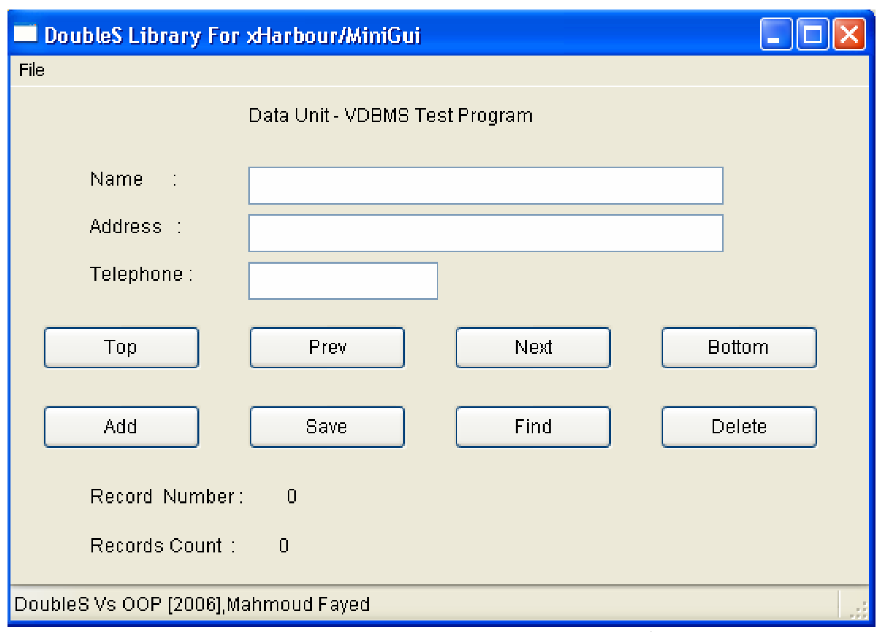


شكل(٣٠) – متطلبات التعريف لكل من الزبون او الخادم عن طريق وحدة النقض Veto Unit

ملحوظة هامة

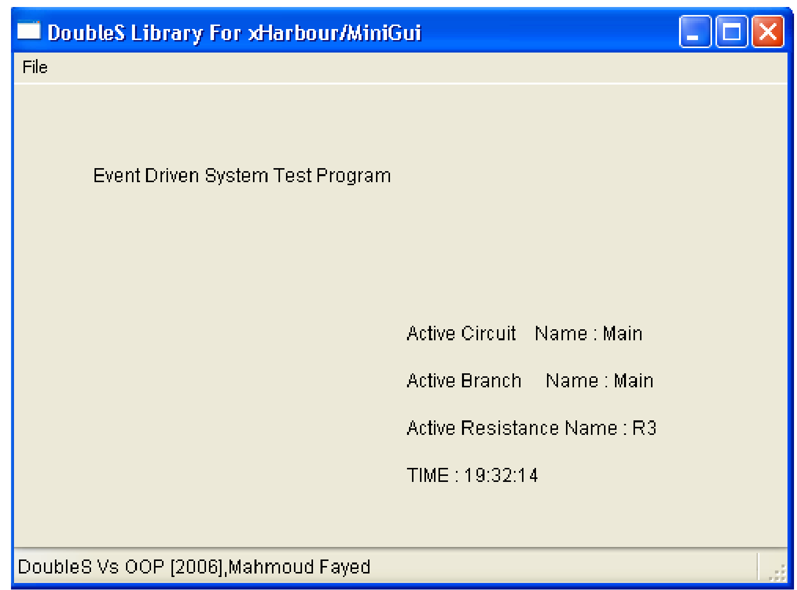
لا تشغل بالك بصعوبة كتابة التعليمات والتعقيد الذى قد يوجد بها – لان محيط التطوير الخاص بنمط البرمجة الخادم الممتاز DoubleS Framework يحل تلك المشكلة ويقدم لك واجهة رسومية سهلة لتصميم الخادم Server.

مثال على وحدة البيانات Data Unit Example :-



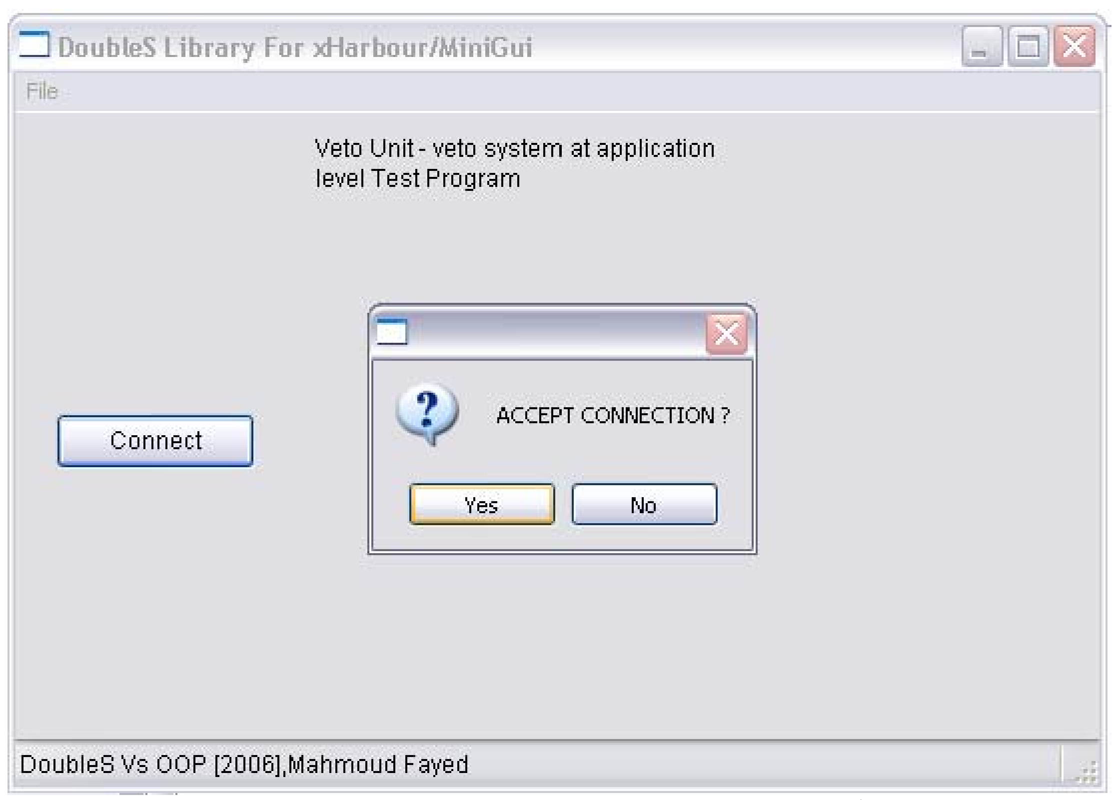
شكل(٣١) – مثال بسيط على وحدة البيانات

مثال على وحدة التعليمات Code Unit Example :-

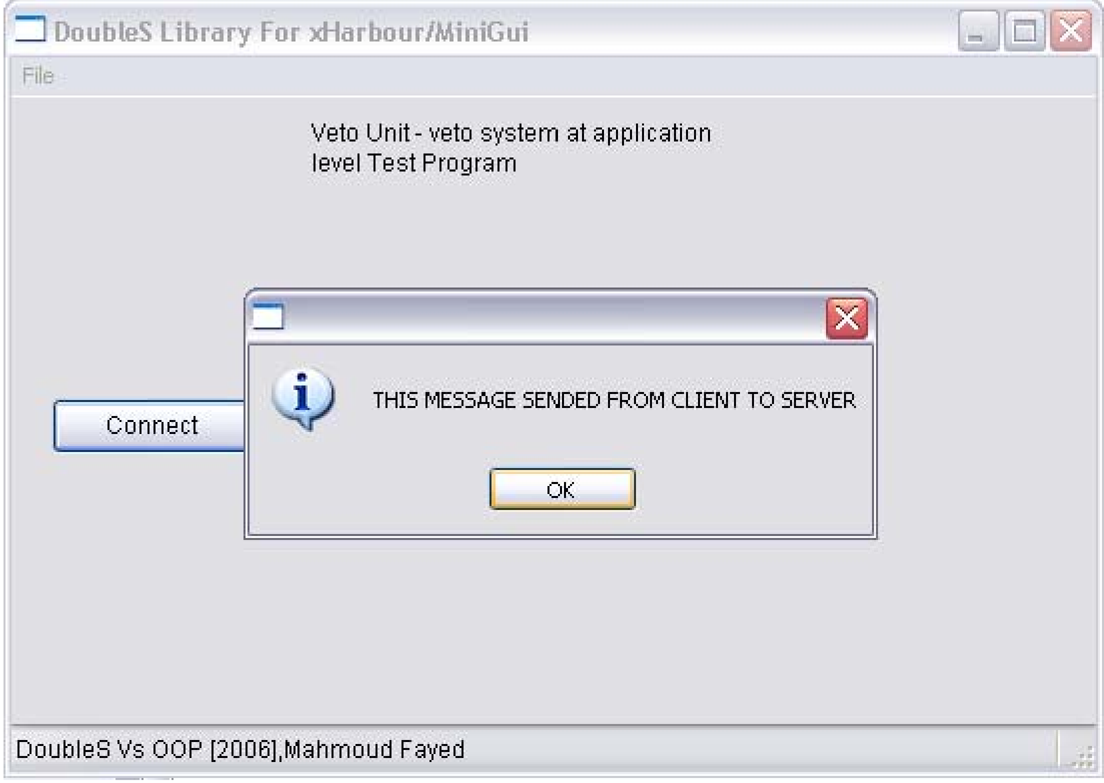


شكل(٣٢) – مثال بسيط على وحدة التعليمات

مثال على وحدة النقض Veto Unit Example :-



شكل(٣٣) – الخادم يستقبل طلب Connection من خادم اخر



شكل(٣٤) – الخادم يستقبل رسالة من خادم اخر

تفاصيل محاكاة برمجة الكائنات :-

يمكن من خلال جمل المقاومات وبالتعاون مع وحدتى البيانات والتعليمات Data Unit & Code Unit عمل محاكاة لنمط برمجة الكائنات

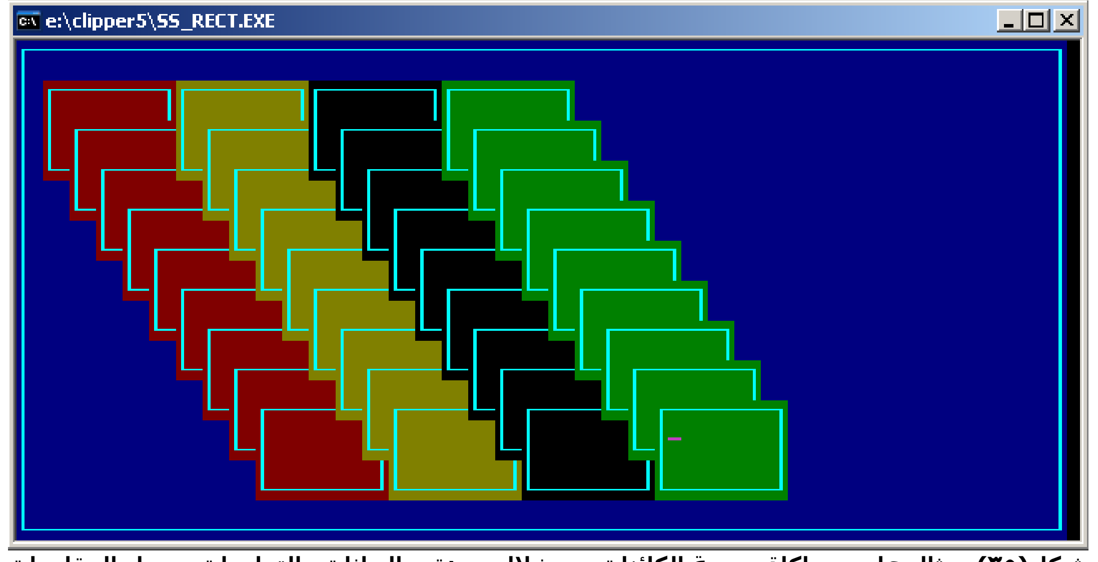
```
CREATE CLASS <X1> [FROM <X2>] DATA <X3> METHODS <X4>  && CREATE NEW CLASS
CREATE OBJECT [<X1>] FROM <X2>                        && CREATE NEW OBJECT
WITH OBJECT <X1>                                      && WITH OBJECT
ENDWITH OBJECT                                        && END WITH OBJECT
DELETE CLASS <X1>                                     && DELETE CLASS
DELETE OBJECT <X1>                                    && DELETE OBJECT
OOP <X1>                                              && invoke method
OOP <X1> = <X2>                                       && assignement
CALLED_AS_METHOD
```

انظر المثال التالى والذى ينشى فصيلة Rectangle ترسم مربع على الشاشة فى المكان المحدد وباللون الذى نريده

```
* For CA-Clipper/xHarbour/xBase++
#INCLUDE "DOUBLES.CH"
DO SSLIB
START DOUBLES
-----------------------------------------------------------------------------*
* This file generated by DoubleS Framework 1.0
* True DoubleS Compiler -> Standard DoubleS Syntax & Statements
* Date : 30/12/2006
* Time : 13:55:23
*-----------------------------------------------------------------------------*
New Server SS_RECT Type Slave Server Eigen Value 000
Details
Data Unit :
        Atom RECT
                Shell K
                        SubShell S
                                Var TOP
                                Var LEFT
                                Var WIDTH
                                Var HEIGHT
                                Var COLOR
                                Var BACKCOLOR
Code Unit :
        Main resistance CIRCUITS\MAIN\MAIN\MAIN
        Circuit       MAIN
                Branch     MAIN
```

```
                ParallelTo 0
                        Resistance MAIN
                        Resistance INIT
                        Resistance DRAW
Veto Unit :
End Of Server
Resistance RDBMVR1() address Code Unit : Circuits\MAIN\MAIN\MAIN
Resistance RDBMVR2() address Code Unit : Circuits\MAIN\MAIN\INIT
Resistance RDBMVR3() address Code Unit : Circuits\MAIN\MAIN\DRAW
Server FireON
* Resistance Code Unit : Circuits\MAIN\MAIN\MAIN
Resistance RDBMVR1() code
CREATE CLASS RECT DATA RECT:K:S METHOD MAIN\MAIN
CREATE OBJECT RECT1 FROM RECT
OOP RECT1.DRAW()
CREATE OBJECT RECT2 FROM RECT
OOP RECT2.WIDTH = 10
OOP RECT2.HEIGHT = 5
OOP RECT2.COLOR = "BG+/R"
OOP RECT2.BACKCOLOR = "BG+/R"
FOR XVAR = 2 TO 18  STEP 2
        OOP RECT2.TOP = XVAR
        OOP RECT2.LEFT = XVAR
        OOP RECT2.DRAW()
NEXT
OOP RECT2.COLOR = "BG+/GR"
OOP RECT2.BACKCOLOR = "BG+/GR"
FOR XVAR = 2 TO 18  STEP 2
        OOP RECT2.TOP = XVAR
        OOP RECT2.LEFT = XVAR + 10
        OOP RECT2.DRAW()
NEXT
OOP RECT2.COLOR = "BG+/N"
OOP RECT2.BACKCOLOR = "BG+/N"
FOR XVAR = 2 TO 18  STEP 2
        OOP RECT2.TOP = XVAR
        OOP RECT2.LEFT = XVAR + 20
        OOP RECT2.DRAW()
NEXT
OOP RECT2.COLOR = "BG+/G"
OOP RECT2.BACKCOLOR = "BG+/G"
FOR XVAR = 2 TO 18  STEP 2
```

```
        OOP RECT2.TOP = XVAR
        OOP RECT2.LEFT = XVAR + 30
        OOP RECT2.DRAW()
NEXT
INKEY(0)
SET COLOR TO W/N
CLEAR
QUIT
End Of Resistance
* Resistance Code Unit : Circuits\MAIN\MAIN\INIT
Resistance RDBMVR2() code
OOP THIS.TOP = 0
OOP THIS.LEFT = 0
OOP THIS.WIDTH = 79
OOP THIS.HEIGHT = 25
OOP THIS.COLOR = "bg+/b"
OOP THIS.BACKCOLOR = "bg+/b"
End Of Resistance
* Resistance Code Unit : Circuits\MAIN\MAIN\DRAW
Resistance RDBMVR3() code
SET COLOR TO (OOP THIS.BACKCOLOR)
@(OOP THIS.TOP) , (OOP THIS.LEFT) CLEAR TO ((OOP THIS.HEIGHT);
( OOP THIS.TOP)-1) ,( (OOP THIS.LEFT) + (OOP THIS.WIDTH)-1)
SET COLOR TO (OOP THIS.COLOR)
@ (OOP THIS.TOP) , (OOP THIS.LEFT) TO ((OOP THIS.HEIGHT) +;
( OOP THIS.TOP)-1) ,( (OOP THIS.LEFT) + (OOP THIS.WIDTH)-1)
End Of Resistance
*------------------------------------------------------------------------------*
```



شكل(٣٥)- مثال على محاكاة برمجة الكائنات من خلال وحدتى البيانات والتعليمات وحجم المقاومات

مثال على خادم الجرافك وخادم الصوت :-

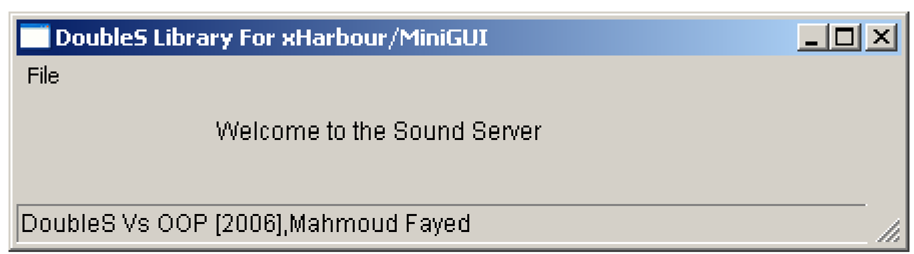
من الامثلة البسيطة التى توضح مفهوم الخادم هما خادم الجرافك وخادم الصوت
ان خادم الجرافك هو عبارة عن خادم يقدم خدمة الرسم على الشاشة
بينما خادم الصوت هو خادم يقدم خدمة تشغيل ملفات الصوت

خادم الجرافك :- تم كتابة خادم لديه
القدرة على الرسم على الشاشة من خلال مكتبة جرافك (هذا الخادم يعمل تحت
نظام Dos القديم – وتم برمجته من خلال نمط
البرمجة الخادم الممتاز DoubleS عن طريق محيط التطوير DoubleS Framework و
المكتبة DoubleS Library ولغة البرمجة
القديمة كليبر (CA-Clipper) ويقدم هذا الخادم خدمة الرسم لاى خادم يستطيع الاتصال
به – وبالفعل تم عمل برنامج اخر عبارة عن زبون Client وهو برنامج بسيط مطور بنمط
البرمجة الخادم الممتاز DoubleS ولكنه يعمل تحت نظام Windows وتم برمجته عن
طريق محيط التطوير DoubleS Framework و المكتبة DoubleS Library ولغة البرمجة
xHarbour/MiniGUI (المترجم xHarbour والمكتبة MiniGUI).

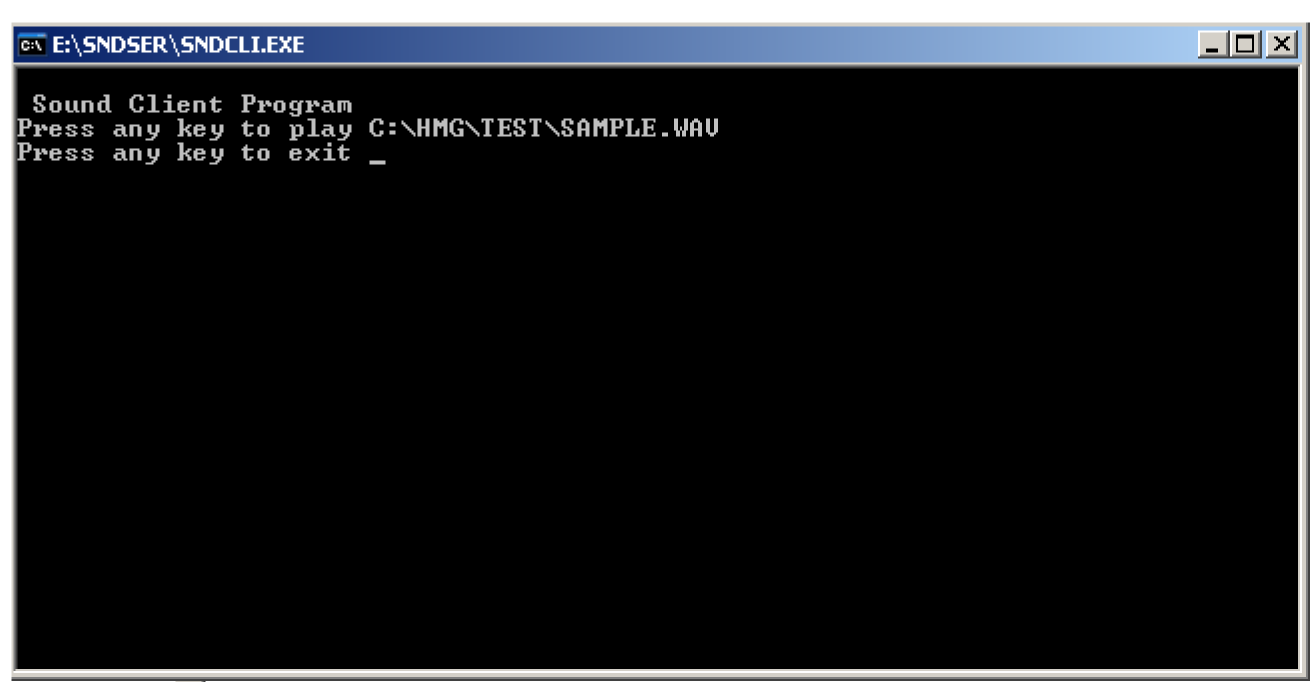
<u>وهنا نلاحظ ان الخادم برنامج يعمل تحت DOS بينما الزبون برنامج يعمل تحت Windows</u>

خادم الصوت:- تم كتابة خادم لديه القدرة على تشغيل ملفات الصوت (هذا الخادم
يعمل تحت Windows وتم برمجته باستخدام نمط البرمجة الخادم الممتاز DoubleS
عن طريق محيط التطوير DoubleS Framework و المكتبة DoubleS Library ولغة
البرمجة xHarbour/MiniGUI (المترجم xHarbour والمكتبة MiniGUI).
وتم عمل الزبون Client الذى يرسل اسم ملف الصوت الى الخادم كى يقوم الخادم
بتشغيله – وتم كتابة الزبون تحت نظام Dos

<u>وهنا نلاحظ ان الخادم برنامج يعمل تحت Windows بينما الزبون برنامج يعمل تحت Dos</u>



شكل(٣٦) – مثال على خادم الصوت



شكل(٣٧) – برنامج الزبون **Client** الذى يستخدم خادم الصوت

وفيما يلى التعليمات الخاصة بخادم الصوت

```
#include "DoubleSHMG.ch"
#include "minigui.ch"
SET PROCEDURE TO SSLIB.PRG
Function Main
START DOUBLES          && ONE TIME AT THE TOP OF DOUBLES
APPLICATION
Set veto system level 2
Set veto system path E:\vetosys
New Server MYSERVER  Type Slave Server Eigen Value 000
Details
   DataUnit :
         Atom CHANNEL
            Shell K
               SubShell S
   CodeUnit :
         Circuit        C1
         MainResistance CONTROL
         MainSwitch     On
            Branch      B1
            SWITCH OFF
               Resistance R1
               Resistance R2
               Resistance R3
   VetoUnit :
```

```
RECEIVING_VETO MYREC
Veto SHOWDATA
    Type General
    Circuit   C1
    Branch    B1
    Resistance R1
Veto PLAYSOUND
    Type General
    Circuit   C1
    Branch    B1
    Resistance R2
Veto MYREC
    Type General
    Circuit   C1
    Branch    B1
    Resistance R1
    Channel MYINPUTCHANNEL
    Type     Input Channel
    Atom     CHANNEL
    Shell    K
    SubShell S
End Of Server
Resistance R1() Address Code Unit : Circuits\C1\B1\R1
Resistance R2() Address Code Unit : Circuits\C1\B1\R2
Resistance CONTROL() Address Code Unit : CONTROL
Select INPUT CHANNEL MYINPUTCHANNEL
  * LOGO SCREEN WINDOW
    DEFINE WINDOW Win_1 ;
    AT 0,0 ;
    WIDTH 450 ;
    HEIGHT 120 ;
    TITLE 'DoubleS Library For xHarbour/MiniGUI' ;
    MAIN      ;
    ON init runserver()  ;
    ON release shutdown()
    DEFINE MAIN MENU
      POPUP "File"
        ITEM 'Exit From Application' ACTION win_1.release
      END POPUP
    END MENU
    DEFINE STATUSBAR
        STATUSITEM "DoubleS Vs OOP [2006],Mahmoud Fayed"
```

```
        END STATUSBAR
        DEFINE LABEL Label0
        ROW 10
        COL 100
        WIDTH 200
        HEIGHT 30
        VALUE "Welcome to the Sound Server "
        END LABEL
   END WINDOW
    CENTER WINDOW Win_1
    ACTIVATE WINDOW Win_1
Return
FUNCTION runserver()
   SERVER FIREON
   WIN_1.RELEASE
   RETURN
FUNCTION CONTROL()
   CHECK_VETO_SERVICES
   DO EVENTS
   RETURN
FUNCTION SHUTDOWN()
   SERVER SHUTDOWN
   RETURN
Resistance R1() CODE
IF REQUEST_TYPE_CONNECTION
   ACCEPT CONNECTION
ENDIF
End Of Resistance

Resistance R2() CODE
OPEN CHANNEL MYINPUTCHANNEL && Select ADDRESS CHANNEL:K:S
                           && LOAD ACTIVE SUBSHELL FROM MEMORY
GOTO FIRST ELECTRON
PLAY WAVE GET_ACTIVE_ELECTRON_VALUE
CLEAR CHANNEL
CLOSE CHANNEL
End Of Resistance
*--------------------------------------------------------------------------------*
```

وفيما يلى التعليمات الخاصة بالزبون Client

```
* For Clipper/Xharbour
#INCLUDE "DOUBLES.CH"  && AT THE START OF EACH SERVER FILE
DO SSLIB
START DOUBLES          && ONE TIME AT THE TOP OF DOUBLES
APPLICATION
*
       Set veto system level 2
       Set veto system path E:\vetosys
*---------------------------------------------------------------------------*
* This file generated by DoubleS Framework 1.0
* True DoubleS Compiler -> Standard DoubleS Syntax & Statements
* Date : 06/20/06
* Time : 21:15:49
*---------------------------------------------------------------------------*
New Server MYCLIENT Type Slave Server Eigen Value 000
Details
   Data Unit :
         Atom Clients
            Shell K
               SubShell S
                  Var myclient
         Atom CHANNEL
            Shell K
               SubShell S
                  Var mychannel
   Code Unit :
   Veto Unit :
      Channel mychannel
            Type     Input Channel
            Atom     CHANNEL
            Shell    K
            SubShell S
     Connection MyClientConnection
            Type          New Client (New Object )
            OutputChannel mychannel
            Atom          Clients
            Shell         K
            SubShell      S
            Electron      myclient
            Server        MYSERVER
End Of Server
```

```
? " Sound Client Program"
? "Press any key to play C:\HMG\TEST\SAMPLE.WAV"
inkey(0)

SELECT CLIENT CONNECTION CLIENTS:K:S:MYCLIENT
CON_CONNECT
CON_SENDDATA "C:\HMG\TEST\SAMPLE.WAV"
CON_SENDVETO PLAYSOUND
CON_DISCONNECT

? "Press any key to exit "
inkey(0)
*-------------------------------------------------------------------------------------*
```

ملحوظة هامة

- بالنسبة لخادم الجرافك فهو متوفر كاحد الامثلة Samples الخاصة بمكتبة الخادم الممتاز DoubleS Library
- يمكن التراسل بين تطبيقات الخادم الممتاز DoubleS بصرف النظر عن المنصة التى يعمل منها التطبيق (Dos او Windows او Linux او OS/2 وهكذا) ولهذا من المكن ان يكون الخادم يعمل تحت منصة معينة – بينما يعمل الزبون تحت منصة اخرى وبالتاكيد ذلك لا يمنع ان يكون كل من الخادم والزبون يعملان تحت نفس المنصة.

تطبيقات الطبقات المتعددة N-Tier Applications :-

تنقسم التطبيقات الى اكثر من طبقة – من المفترض ان تكون كل طبقة مستقلة عن الاخرى من حيث امكانية التعديل فى محتوياتها بدون الحاجة الى المساس بالطبقات الاخرى الا عند الضرورة القصوى – والطبقات التى يمكن ان يتكون منها التطبيق مختلفة مثل طبقة البيانات وطبقة المنطق وطبقة واجهة النظام وطبقة الاتصالات وغيرها حسب حاجة التطبيق.

ان نمط البرمجة هو العنصر الاساسى فى طبقة المنطق Bussinness Logic الخاصة بالتطبيق – واذا تخيلنا احد التطبيقات التجارية كمثال للتطبيق الذى نطوره – فاننا نبدا بتصميم قاعدة البيانات – ثم واجهة النظام و منطق التطبيق

وقد يكون التطبيق الذى نطوره عبارة عن Desktop Application وعندها توجد البيانات والواجهة ومنطق التطبيق معا على جهاز واحد – وقد يكون التطبيق عبارة عن -Client Server كان تكون قاعدة البيانات SQL Server او Oracle قد تم تحميلها على خادم

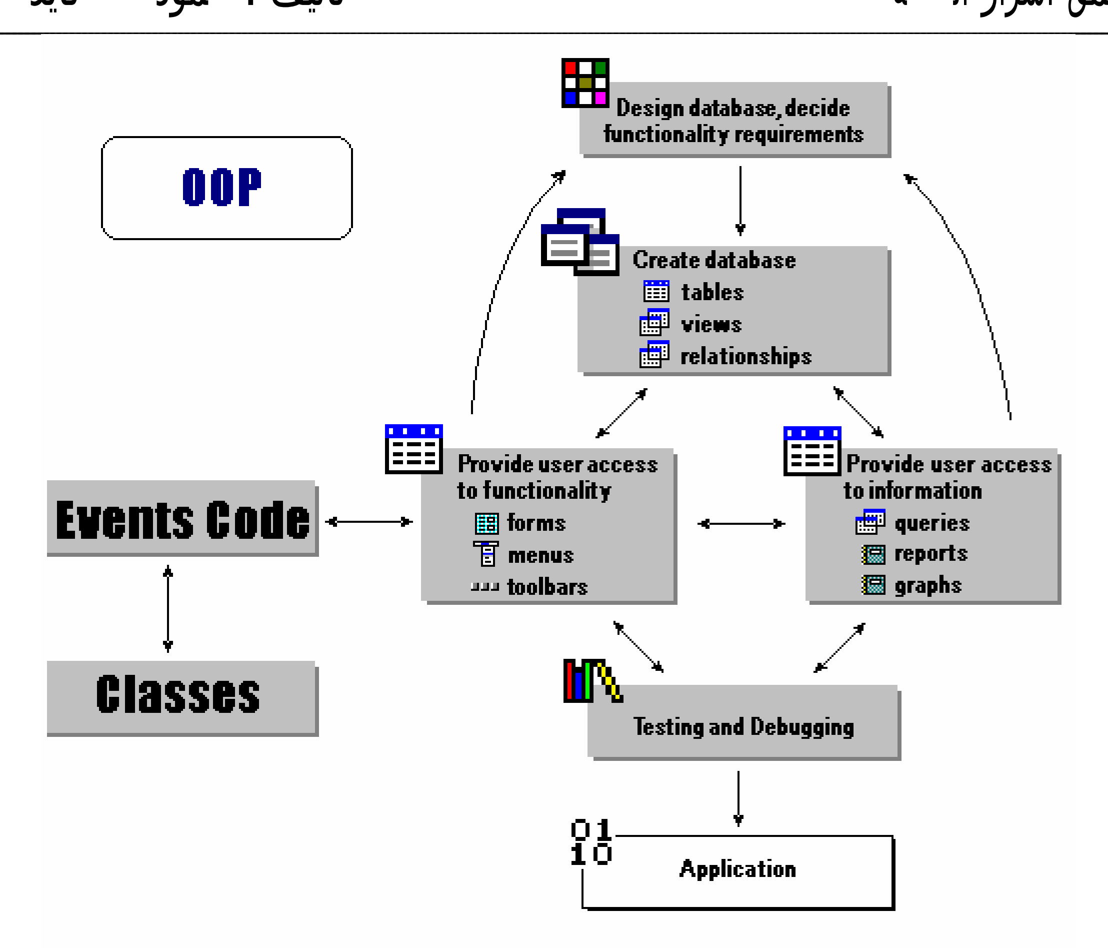
موجود على الشبكة وعندها تتواجد البيانات على جهاز بينما التطبيق على جهاز اخر ونقتصر نحن على تطوير التطبيق الذى يكون عبارة عن Client لان الخادم Server هو موجود بالفعل – وقد يكون التطبيق الذى نطوره عبارة عن Distributed Application اى تطبيق موزع – وفى هذه الحالة قد تكون البيانات موزعة على اكثر من جهاز او متواجدة فى مكان واحد بينما يتم توزيع كل من واجهة التطبيق ومنطق البرنامج الى اكثر من جزء منفصل يعمل على حده وقد نستخدم لذلك تقنيات مثل COM/DCOM/COM+ والجدير بالذكر ان تقنية ال COM اصبحت قديمة الان وخصوصا فى عالم الدوت نت NET Framework.

الخاص بشركة مايكروسوفت – بينما يظل +COM مستخدما وخصوصا مع صفحات ASP.NET وقد يكون التطبيق الذى نطوره عبارة عن WEB APPLICATION وهنا قد يوجد للبرنامج اكثر من واجهة مثل واحدة تحت نظام

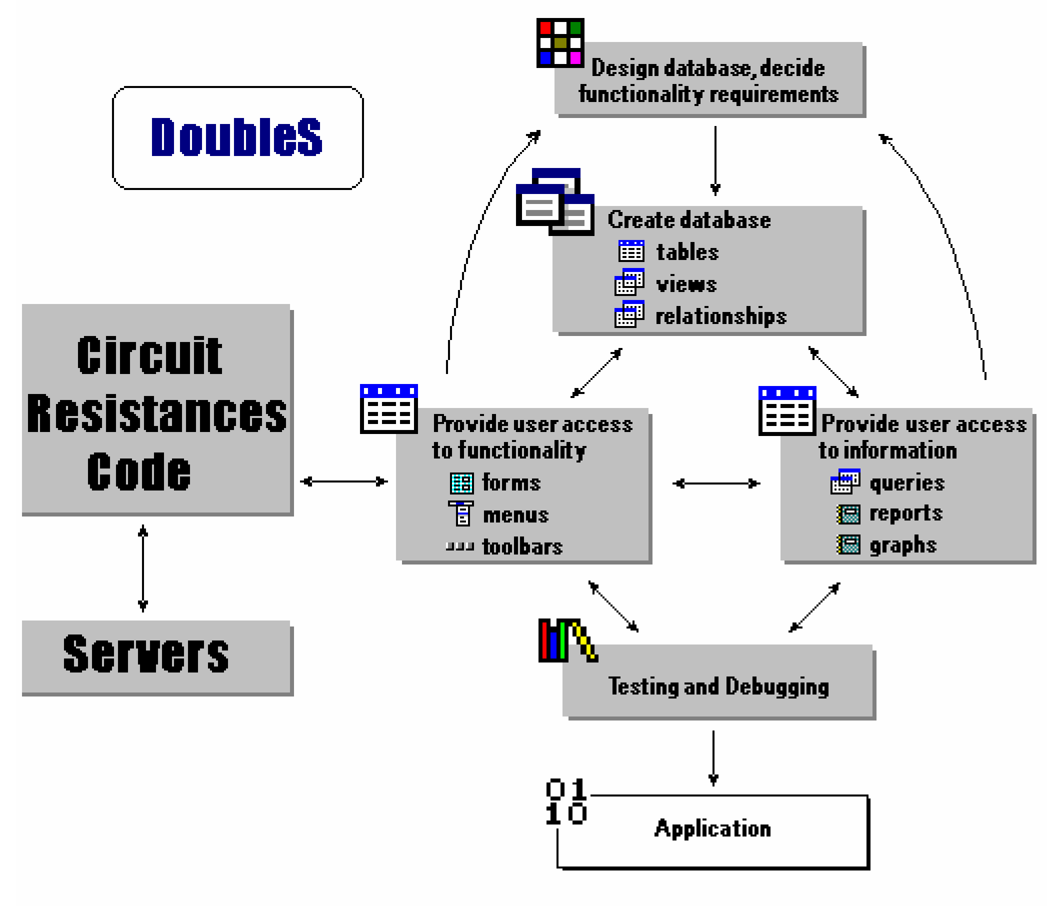
Windows اى Windows Forms وواحدة من خلال الويب Web Form مثل اى موقع وواحدة للموبايل Mobile Application

ان نمط البرمجة الخادم الممتاز DoubleS يتواجد فى هذا العالم المثير من التطبيقات – لكى يقوم بدور كفء كنمط برمجة مثالى بديل لنمط برمجة الكائنات – لهذا كل ماعليك ان تتخيله فى عالم التطبيقات المتعددة هو ان تستبدل نمط برمجة الكائنات بنمط البرمجة الجديد الخادم الممتاز DoubleS

انظر شكل (٣٨) الذى يوضع دور نمط برمجة الكائنات – ثم انظر شكل (٣٩) لترى ان نمط البرمجة الخادم الممتاز قد اصبح بديلا لنمط برمجة الكائنات.



شكل(٣٨) – تصميم التطبيقات التجارية من خلال نمط برمجة الكائنات

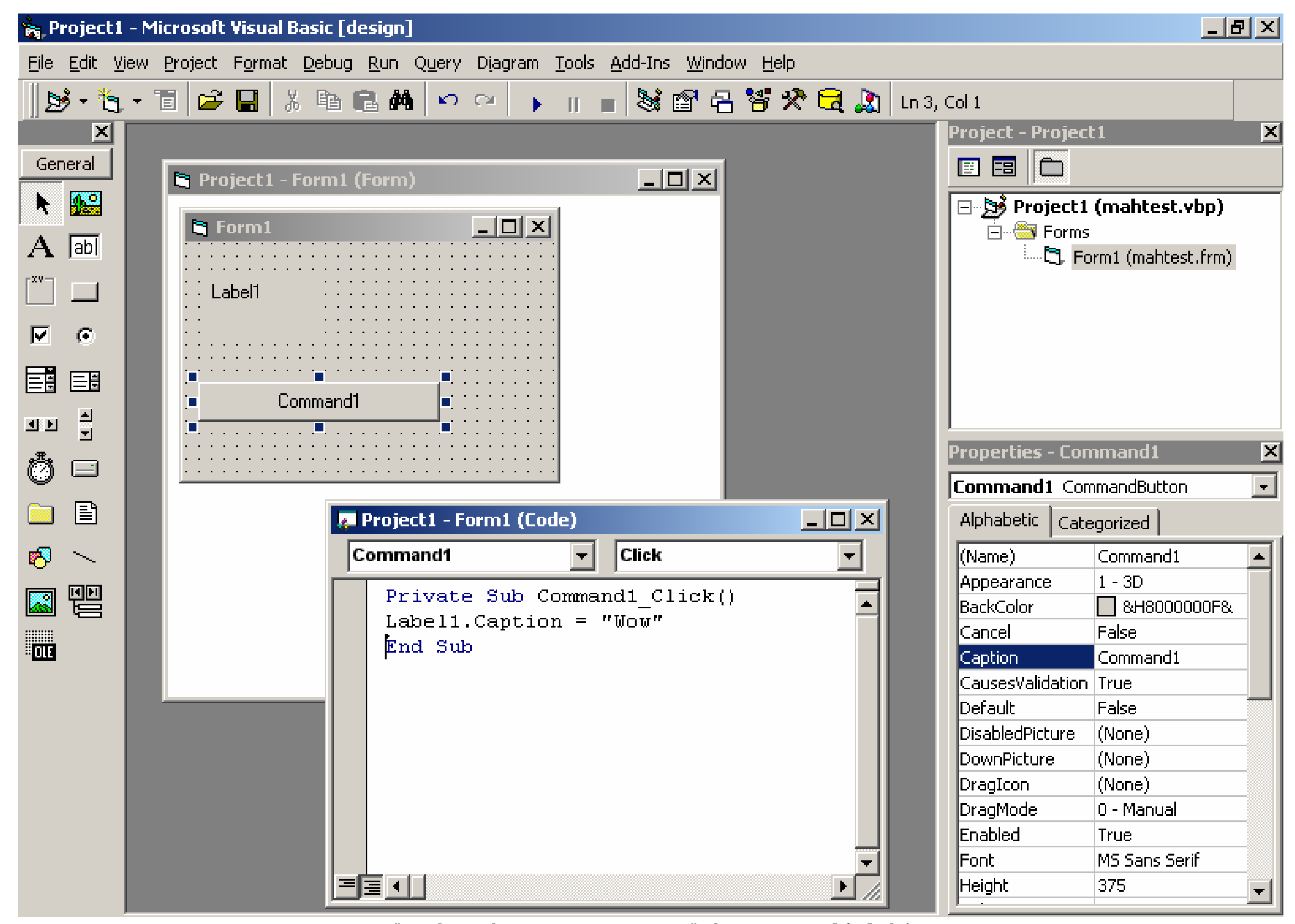


شكل(٣٩) – تصميم التطبيقات التجارية من خلال نمط البرمجة الخادم الممتاز

مفهوم الخادم كمترجم Server As Compiler :-

ان برمجة الكائنات ترتكز فى اداء اى عملية Process على الطرق Method بمعنى ان تعديل مواصفات اى خاصية للكائن Attribute لا يظهر تاثيرها الا اذا تم نداء طريقة Method وليكن مثلا Refresh() – مع العلم بان هناك شواذ لتلك القاعدة.

انظر المثال التالى – باستخدام اللغة Visual Basic 6 والذى يبين ظهور التاثير مباشرة بدون الحاجة لاستدعاء طريقة



شكل(٤٠)- تنفيذ عملية بمجرد تغير محتويات خاصية

فى الواقع ان ظهور التاثير المباشر لتغير خاصية بدون استدعاء طريقة Method يعد نوع من التحايل اما بالنظر الى التعليمات التالية لاستخدام ADO مع قاعدة البيانات SQL Server فاننا نجد انه رغم ضبط خصائص الكائن فانه لايتم تنفيذ اى شى حتى يتم استدعاء طريقة Method وهذا هو الوضع الطبيعى فى برمجة الكائنات.

```
Dim mycon As New ADODB.Connection
im myrec As New ADODB.Recordset
mycon.ConnectionString = "Provider=SQLOLEDB.1;Integrated
Security=SSPI;Persist Security Info=False;Initial Catalog=mahtel;Data
Source=FAYEDCOM\MAHSQL"
```

```
mycon.Open
myrec.Open "select * from tel", mycon
Label1.Caption = myrec.Fields.Item(0).Value
Dim x As Integer
List1.Clear
Do Until myrec.EOF
     List1.AddItem (myrec.Fields.Item(0).Value)
     myrec.MoveNext
Loop
myrec.Close
mycon.Close
```

يختلف الامر فى نمط برمجة الخادم الممتاز الذى يرتكز على الخادم بدلا من الفصيلة – حيث يمكن تنفيذ عمليات بمجرد استقبال بيانات Data ولا يشترط ان يتم استقبال Veto حتى يتم تنفيذ العملية – ان هناك فصل بين البيانات Data والعمليات.

لذا من الممكن ان يعمل الخادم كمترجم Compiler اذا عمل على تنفيذ عمليات بناء على البيانات وكانت هذه البيانات هى عبارة عن تعليمات – وحتى تكون الجملة سليمة من الناحية العلمية فاننا نقول ان الخادم يعمل كمفسر Interpreter وليس كمترجم لانه هنا لا يحول التعليمات الى لغة الالة وانما يقوم بتنفيذها اثناء وقت التشغيل.

ماذا بعد الخادم الممتاز :-

سوف يتم استخدام هذا النمط فى عمل لغة برمجة جديدة تعتمد على مفهوم البرمجة بدون اكواد حيث يكون البديل للتعليمات والكود هو مصمم الهدف Goal Designer الذى سوف يرتكز على العديد من الخوادم (الالاف من الخوادم التى تعمل كمترجم).

DoubleS كقاعدة للعديد من الابحاث العلمية

انه نمط برمجة جديد لذلك يفتح المجال للعديد من الابحاث العلمية – على سبيل المثال

- DoubleS Paradigm & Database Applications
- DoubleS Paradigm & OOP Systems developing
- DoubleS Paradigm & AOP Systems developing
- DoubleS Paradigm & Graphics Applications
- DoubleS Paradigm & Information Systems
- DoubleS Paradigm & Games Programming
- DoubleS Paradigm & Systems Programming

- DoubleS Paradigm & Event Driven Programming
- DoubleS Paradigm & Complex Data Structure
- DoubleS Paradigm & Client-Server Programming
- DoubleS Paradigm & Distributed Applications
- DoubleS Paradigm & Embedded Systems Programming
- DoubleS Paradigm & Grid Computing Programming
- ....And More!

DoubleS كبنية اساسية للغات البرمجة المتطورة

ان الطريق الصحيح يظهر وينتشر معا الوقت وقريبا سوف يكون نمط البرمجة الخادم الممتاز متوفرا فى جميع لغات البرمجة المتطورة والتى سوف تاخذ لقب لغات السوبر Super Languages على سبيل المثال Super C و Super Basic و Super FoxPro و Super Delphi و Super Java و Super xHarbour و هكذا...

كيفية المشاركة فى هذه الثورة العلمية

يتوقف ذلك حسب مهارتك فى البرمجة وخبراتك التى قد اكتسبتها – ابد الان بالحصول على DoubleS مجانا من خلال الموقع
http://www.sourceforge.net/projects/doublesvsoop ثم قم ايضا بالحصول على نسختك من اللغة المجانية xHarbour/MiniGUI من خلال الموقع
http://www.sourceforge.net/projects/harbourminigui
ثم تعلم DoubleS جيدا ويمكنك الاطلاع على الشفيرة المصدرية الخاصة بكل من DoubleS Framework و DoubleS Library

لذلك يمكنك المساهمة كالتالى :-

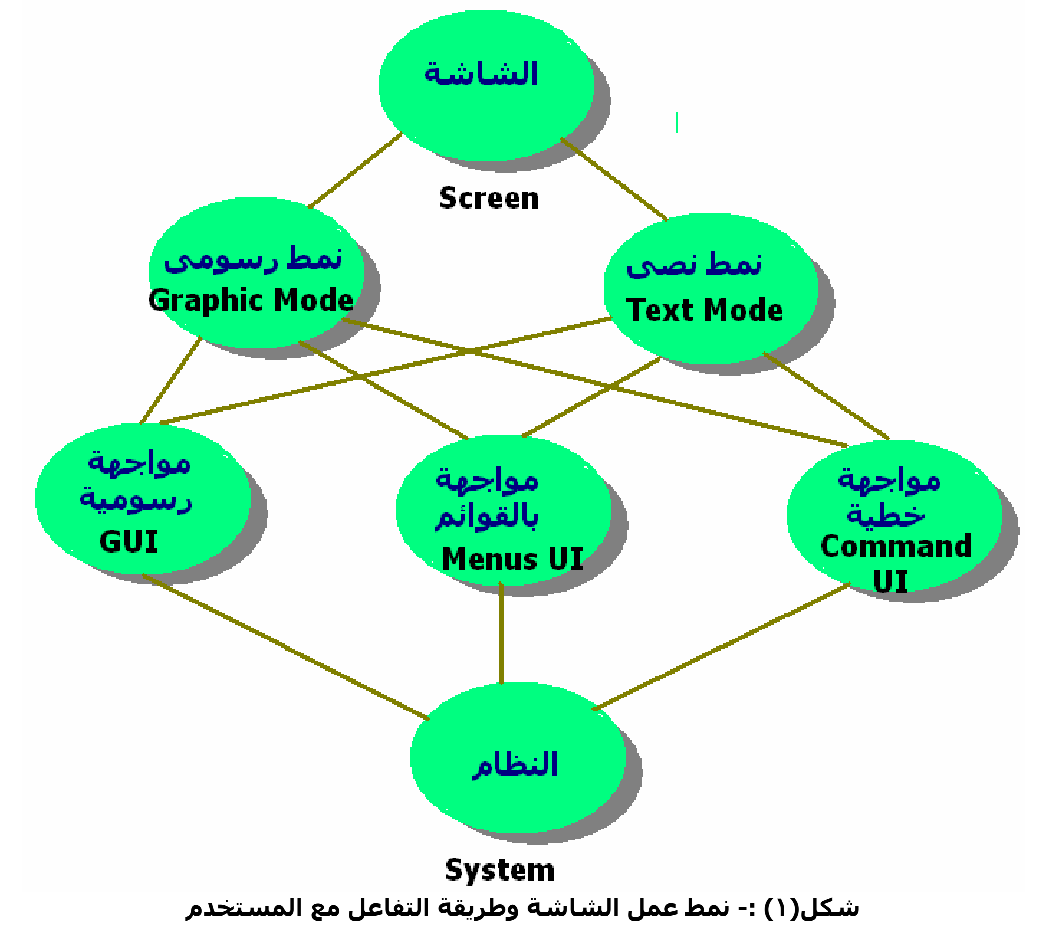
- تطوير DoubleS Framework
- تطوير DoubleS Library
- نقل الـ DoubleS الى لغات البرمجة الاخرى
- امداد مستخدمى DoubleS بالعديد من الامثلة Samples والتطبيقات Applications التى صممتها به
- كتابة المقالات والكتب التى تخدم DoubleS
- العمل على ابحاث Researches تساهم فى تطوير ونشر DoubleS

الباب الثالث
واجهة النظام

مقدمة هامة :-

ان واجهة النظام System User Interface عنصر هام جدا من عناصر النظام ومع ذلك فان هناك نظم ليس لها واجهة على الاطلاق مثل الانظمة المختبئة داخل نظم اخرى Embedded Systems – عندما يتم تحديد هل سوف تكون هناك واجهة للنظام ام لا – ينبغى ايضا تحديد ماهى ملامح هذه الواجهة اى سماتها والتى ترتكز على شيئين هما

١ – النمط الذى تعمل فيه الشاشة Screen Mode
وهو اما نمط نصى Text Mode او نمط رسومى Graphic Mode
٢ – كيفية تفاعل النظام مع المستخدم how Program Interaction with user وهو اما عن طريق المواجهة الخطية Command Line او من خلال القوائم التى يتم اختيار احد عناصرها من خلال الاسهم و مفتاح الادخال وتسمى User Menus Interfacre واخيرا المواجهة من خلال النوافذ والايقونات والقوائم والفارة ويطلق عليها المواجهة الرسومية GUI وينبغى ادارك ان طريقة التفاعل مع المستخدم تعمل فى اى نمط فمثلا نجد ان المواجهة الخطية Command Line من الممكن ان تعمل فى نمط نصى Text Mode كما يحدث غالبا – او تعمل فى نمط رسومى Graphic Mode – وبالمثل المواجهة الرسومية GUI من الممكن ان تعمل قى نمط رسومى Graphic Mode كما يحدث غالبا – او فى نمط نصى Text Mode – وبالمثل المواجهة بالقوائم Menus والتى تنتشر على حد سواء فى كل من النمط النصى والنمط الرسومى.
انظر شكل (١) والذى يوضح ذلك



شكل(١) :- نمط عمل الشاشة وطريقة التفاعل مع المستخدم

مثال على مواجهة خطية تعمل فى نمط نصى نظام DOS عندما يكون هو النظام المثبت على الحاسب كنقطة بداية – وكمثال على نظام مواجهة خصية يعمل فى نمط رسومى هو نظام DOS ايضا ولكن عندما تقوم بتشغيله من خلال WINDOWS بحيث يعمل فى نافذة من نوافذ Windows – وكمثال على مواجهة رسومية GUI تعمل فى نمط رسومى Graphic Mode هو نظام Windows – وكمثال على مواجهة رسومية GUI تعمل فى نمط نصى Text Mode فهناك العديد من برامج DOS مثل برنامج EDIT المسئول عن تحرير الملفات.

مستويات العمل عند برمجة واجهة النظام :-

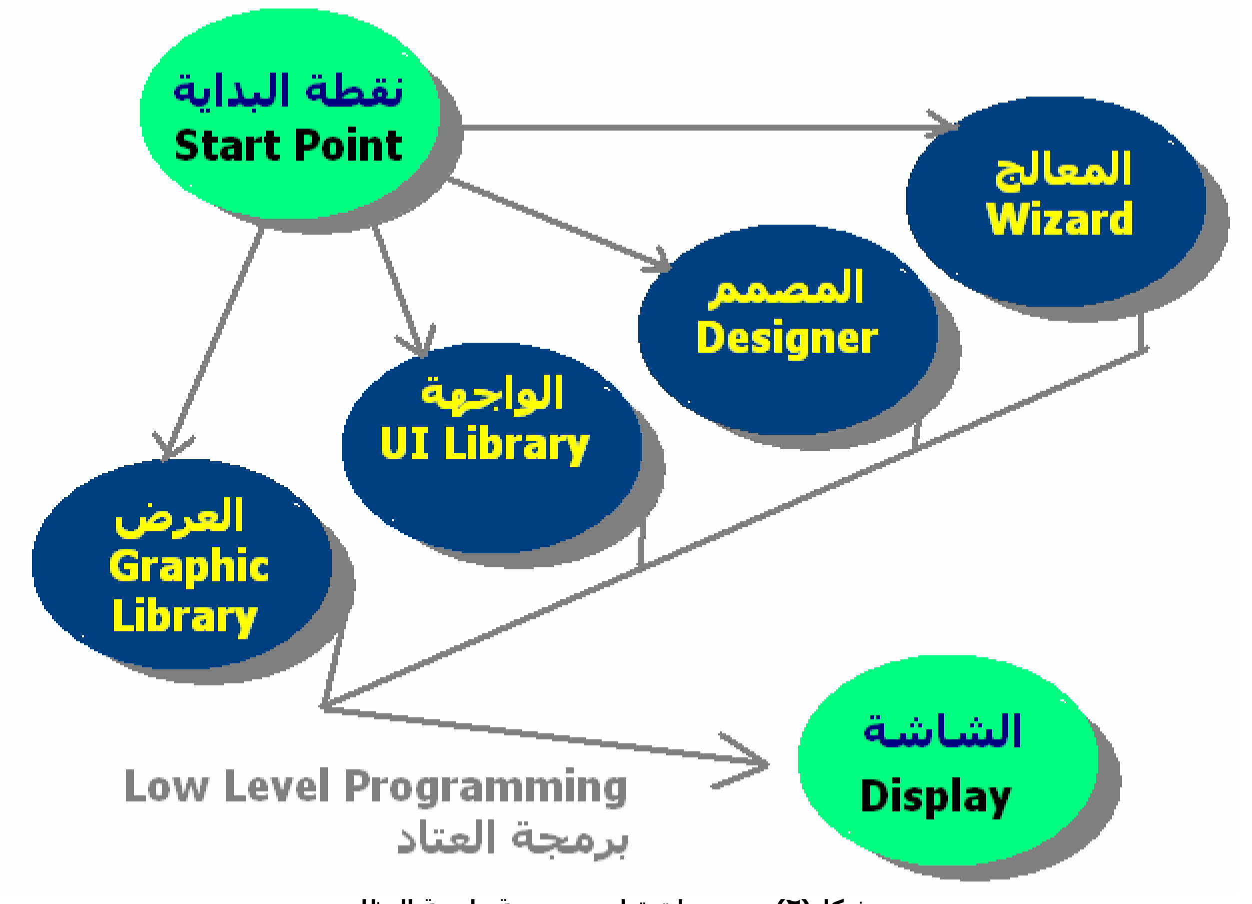
هناك العديد من المستويات

- برمجة واجهة النظام من خلال التعامل المباشر مع العتاد Low Level Programming
- برمجة واجهة النظام من خلال استخدام دوال جاهزة للعرض او الرسم على الشاشة
  - قد تكون هذه الدوال قمنا بكتابتها مسبقا
  - قد تكون متوفرة من قبل لغة البرمجة
  - قد تكون خدمات متاحة من قبل نظام التشغيل
- برمجة واجهة النظام من خلال استخدام مكتبة CUI او MUI او GUI
  - قد تكون هذه الدوال قمنا بكتابتها مسبقا
  - قد تكون متاحة من قبل لغة البرمجة
    - عندها قد تتعامل مع العتاد مباشرة Low Level Programming
    - قد تستند على خدمات من قبل نظام التشغيل
- برمجة واجهة النظام من خلال استخدام ادوات تصميم Screen/Form Designer
  - قد تكون هذه الادوات قمنا بكتابتها مسبقا
  - قد تتوفر فى لغة البرمجة
- برمجة واجهة النظام من خلال استخدام المعالج Wizard ومولد الكود او التعليمات Code Generator

ملحوظة هامة

لابد من ان ترجع المستويات العليا الى المستويات السفلى لكى تنجز المهام الخاصة بها – فمثلا عند عمل نموذج من خلال المعالج Wizard فانه من الاحرى ان ينتج ملف نموذج Form File يستطيع مصمم النماذج Form Designer او مصمم الشاشات Screen Designer (فى حالة Text mode) التعامل معه.
بينما يفترض ان يولد مصمم النماذج ملف تعليمات او اكواد Source Code File يشتمل على تعليمات تستند على مكتبة الواجهة سواء كانت MUI او GUI .
بينما من المفترض ان تستند مكتبة MUI او GUI على مكتبة العرض او الرسم Graphic Library والتى بدورها تستند على التعامل مع المكونات المادية Low Level Programming .

واثناء المرور بهذه المراحل الخمسة – من المحتمل ان يتدخل كل من نظام التشغيل – او لغة البرمجة كوسيط. بين اى مرحلتين من هذه المراحل.



شكل(٢)- مستويات تطوير وبرمجة واجهة النظام

وغالبا مايختص نظام التشغيل بجزء Low Level Programming واحيانات يوفر Graphic Library و UI والذى من الممكن ان توفره لغة البرمجة بجانب توفير Wizard و الـ Designer .

وكملاحظة اخيرة ان المعالج Wizard و المصمم Designer يظهران فقط فى مرحلة تطوير وبرمجة النظام – بينما لا يتواجدان اثناء عمل النظام.

المستوى الاول لبرمجة واجهة النظام Low Level Programming :-

يمكن التعامل مع الشاشة مباشرة من خلال برمجة العتاد – ونحتاج لذلك عند برمجة نظم تشغيل الكمبيوتر Computer Operating System او فى حالة عدم توفر دوال ومكتبات مباشرة تقوم بذلك – او عندما نود ان نقوم بعمل مكتبات وادوات تطوير خاصة بنا.

سوف ناخذ مثال على ذلك – برنامج صغير وبسيط مكتوب بلغة التجميع (الاسمبلى) – هذا البرنامج عبارة عن برنامج BOOT اى مدخل ونقطة انطلاق واقلاع نظام التشغيل – والذى عند ترجمته نحصل على ملف COM سعته 512 BYTE يجب ان يتم

كتابتهما على القرص فى head 0 track 0 Sector 1 - وبالتاكيد سوف يتم اختبار ذلك من خلال قرص مرن Floppy Disk لانه من الخطا الفادح ان يتم عمل ذلك على القرص الصلب. وحتى لانقوم باعادة التشغيل كى نقوم باختبار برنامج الـ BOOT فقد تم استخدام برنامج Virtual PC من اجل ذلك.
انظر شكل (٣). ونتيجة تشغيل برنامج BOOT نحصل على شكل (٤).

شكل(٣) – برنامج Microsoft Virtual PC تحت Microsoft Windows XP



شكل(٤) – مثال على برنامج الـ BOOT

وبرنامج الـ BOOT كالتالى :-

```
.386

_text SEGMENT PUBLIC USE16
assume CS:_text, DS:_text
org 0h

MahmoudOS:

        mov ax, 1301h
        mov bx, 0007h
        mov cx, 23
        mov dh, 23
        mov dl, 0
        push cs
        pop es
        mov bp, String
        int 10h

        mov ax, 1301h
        mov bx, 0007h
        mov cx, 23
        mov dh, 24
        mov dl, 0
        push cs
        pop es
        mov bp, Wow
        int 10h


String = $ + 7C00h
Wow = $ + 7C17h
db "Starting MahmoudOS 2007"
db "Wow, I Love you !      "

ORG 510
DW 0AA55h
_text ENDS

END MahmoudOS
```

ويتم ترجمته كالتالى

ML /AT BOOTSEC.ASM

حيث ان BOOTSEC.ASM هو اسم الملف – و الاختيار AT/ كى ينتج ملف COM
وتمت عملية الترجمة باستخدام MASM 6.1

والان نحن بحاجة الى كتابة برنامج الـ BOOT فى المكان المحدد على القرص.
ولعمل ذلك تم كتابة برنامج خاص بهذه المهمة باستخدام لغة سى وهذا البرنامج كالتالى

```
#include <stdio.h>
#include <conio.h>
#include <bios.h>
#include <dos.h>
#include <stdlib.h>

char __far diskbuf[512];

void main( int argc, char *argv[] )
{
    unsigned status = 0, i;
    struct _diskinfo_t di;
    struct _diskfree_t df;
    unsigned char __far *p, linebuf[17];

        FILE *fp;
        int x;
        fp = fopen("bootsec.com","rb");
        for(x = 1 ; x <= 512 ; x++ )
                diskbuf[x-1] = fgetc(fp);
        fclose(fp);

    if( argc != 5 )
    {
printf( "  SYNTAX: DISK <driveletter> <head> <track> <sector>"
);
    exit( 1 );
    }

    if( (di.drive = toupper( argv[1][0] ) - 'A' ) > 1 )
    {
    printf( "Must be floppy drive" );
```

```
    exit( 1 );
    }
    di.head      = atoi( argv[2] );
    di.track     = atoi( argv[3] );
    di.sector    = atoi( argv[4] );
    di.nsectors = 1;
    di.buffer    = diskbuf;

    /* Get information about disk size. */
    if( _dos_getdiskfree( di.drive + 1, &df ) )
    exit( 1 );

    /* Try reading disk three times before giving up. */
    for( i = 0; i < 3; i++ )
    {
       status = _bios_disk( _DISK_WRITE, &di ) >> 8;
    if( !status )
       break;
    }

    /* Display one sector. */
    if( status )
    printf( "Error: 0x%.2x\n", status );
    else
    {
for( p = diskbuf, i = 0; p < (diskbuf + df.bytes_per_sector); p++ )
    {
       linebuf[i++] = (*p > 32) ? *p : '.';
       printf( "%.2x ", *p );
       if( i == 16 )
       {
          linebuf[i] = '\0';
          printf( " %16s\n", linebuf );
          i = 0;
       }
    }
    }
    exit( 1 );
}
```

ويتم ترجمة البرنامج كالتالى

```
CL BOOTW.C
```

حيث ان BOOTW.C هو اسم ملف البرنامج – وسوف يتنج من الترجمة BOOTW.EXE والذى يتم استخدامه كالتالى

BOOTW A 0 0 1

والذى سوف يقوم بكتابة الملف BOOTSEC.COM على القرص المرن المتواجد فى المحرك A

ملحوظة هامة

يمكنك نقل ملف ال BOOTSEC.COM على القرص المرن باستخدام اى برنامج Disk Editor تفصله – ومع ذلك فقد تم كما سبق عرض برنامج مخصص لهذه المهمة بلغة سى – اذا لم يكن متوفرا لديك برنامج Disk Editor مناسب.

سوف ناخذ الان مثال اخر على التعامل المباشر مع الشاشة – وهذا المثال بلغة سى ويقوم بعرض الحروف من A الى Z حيث يقوم بعرض الحرف A متكرار فى جميع اسطر واعمدة الشاشة ثم ينظر ضغط اى مفتاح حتى يكرر نفس العملية مع الحرف B ثم C وهكذا حتى نصل الى الحرف Z.

وتعليمات البرنامج كالتالى

```
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <dos.h>

void main(void)

{
        char far *ptr ;

        int i,m ;


        for (m=65;m<65+26;m++)

        {

        ptr = (char far *) 0xb8000000 ;

        for (i=0;i<80*50;i++)

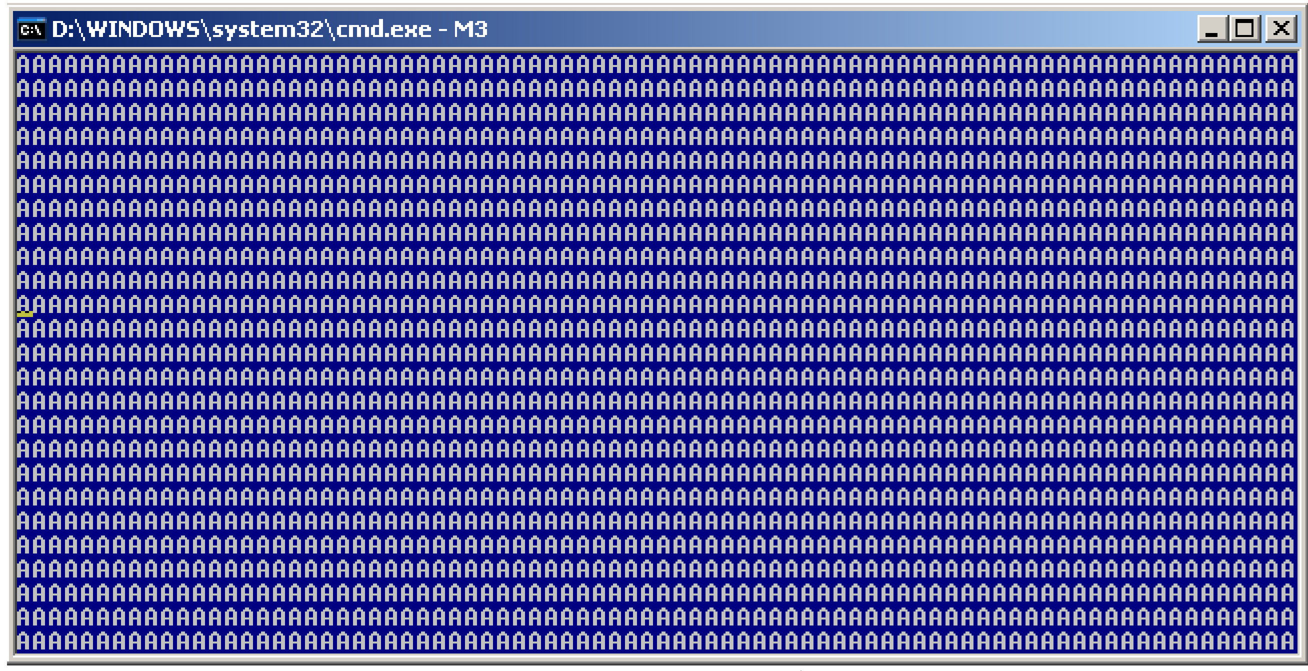
        {
```

```
         *ptr = m ;
         *(ptr+1)=0x17 ;
         ptr+=2 ;

        }
        getch() ;

        }
        exit(0) ;
}
```

ونتيجة تنفيذ البرنامج كما هى بشكل (٥) حيث نجد الحرف A مكررا فى جميع اسطر واعمدة الشاشة.



شكل(٥) – برنامج يتعامل مباشرة مع الشاشة – بلغة سى.

ولا يقف الامر عند التعامل مع الشاشة فقط من اجل توفير واجهة للنظام – بل يمتد ليشمل وحدات الادخال مثل لوحة المفاتيح Keyboard او الفارة Mouse – والمثال التالى بلغة سى يتعامل مع الفارة من خلال Interrupts (مقاطعة المعالج CPU) الخاصة بالفارة – والتى تشترط ان يكون هناك Mouse Driver قد تم تحميله من قبل.

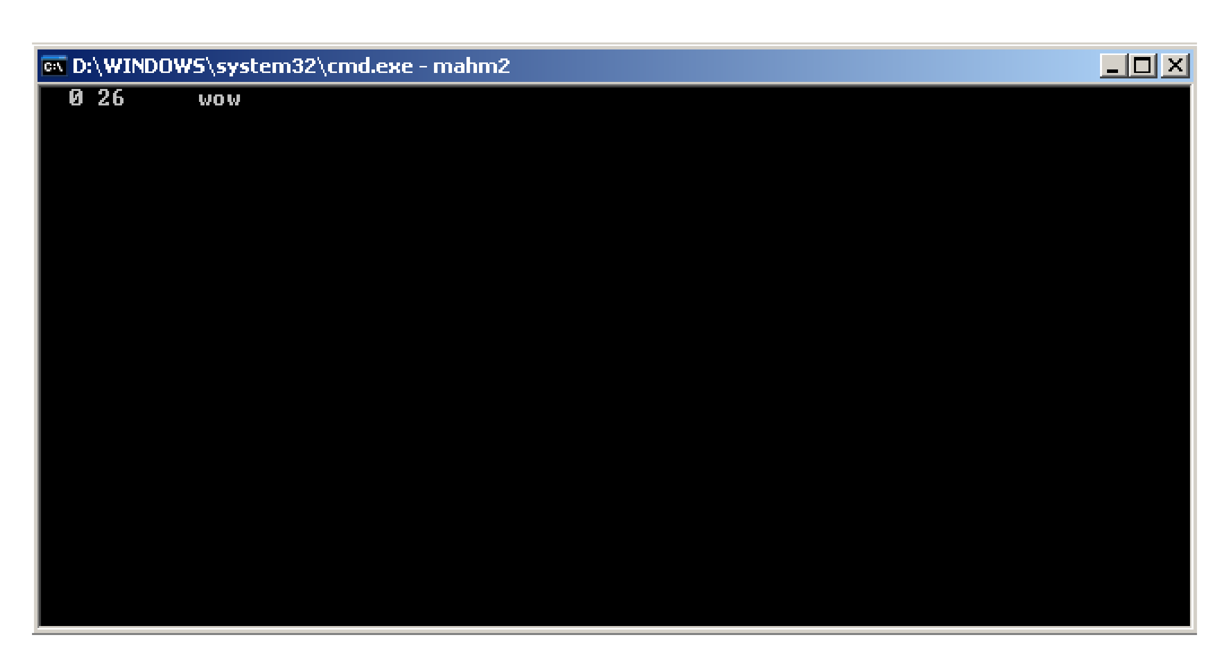
```
#include <stdio.h>
#include <stdlib.h>
#include <dos.h>
#include <conio.h>
#include <graph.h>
```

```
int mx,my;
void main(void)
{
  _clearscreen( _GCLEARSCREEN );
  _asm {
        mov ax,0
        int 33h
        mov ax,1
        int 33h
        mov   ax,1Ch                ; FUC 1Ch: Set mouse interrupt rate
        mov   bx,1
        int   33h
  }
  do {
     _asm{
          mov ax,03
         int 33h
          shr dx,1
          shr dx,1
          shr dx,1  ; myltiply dx by 8

          mov [my],dx
          shr cx,1
          shr cx,1
          shr cx,1  ; myltiply dx by 8
          mov [mx],cx
     }

     _settextposition(0,0);
     printf(" %d %d     wow ",my,mx);
  } while( ! kbhit() ) ;
  _asm {
  mov ax,01
  int 33h
  }
}
```

وهذا البرنامج يستدل على مكان الفارة باستمرا – يظهر رقم السطر Row والعمود على الشاشة.



شكل(٦) – برنامج يتعامل مع الفارة بلغة سى

ولعلك قد تتسال الان عن التعامل مع الصورة – العمل فى Graphic Mode يتم ذلك من خلال اوامر Interrupts خاصة بـ VESA (Video Electronics Standards Association) والتى ان كان يدعمها كارت الشاشة – فانه تتيح التعامل معه بصورة قياسية المثال التالى يقوم بعرض صورة TRUE COLOR وبالتحديد 32 BIT باستخدام Vesa – حيث يقوم البرنامج بفتح ملف BMP (وهو من ابتكار مايكروسوفت وتم نقله الى العديد من الانظمة) به صورة خجمها 800 X 600 ثم يقوم بعرض هذه الصورة على الشاشة – البرنامج تم كتابه ايضا بلغة سى وقد كان هذا البرنامج فى الاصل عبارة عن مثال ياتى مع كتب VESA 3 يشرح كيفية العمل مع الـ Vesa فى نمط شاشة 256 color وقام المولف بعمل التعديلات الازمة لكى يعمل فى True Color وحتى يقوم بقراءة ملف الصورة BMP وعرضه – وهذا مجرد مثال لا يهدف الى ان يكون مكتبة جرافك تستخدم فى التطبيقات العملية

```
/* Original code contributed by: - Kendall Bennett, SciTech Software
 Conversion to Microsoft C by: - Rex Wolfe, Western Digital Imaging
 Hicolor modes & BMP READ by : - Mahmoud Fayed, Electronics Engineering
Faculty */

#include <stdio.h>
#include <stdlib.h>
#include <dos.h>
#include <conio.h>

#define DIRECT_BANKING
#ifdef DIRECT_BANKING
extern far setbxdx(int, int);
```

```
#endif
/*--------------------- Macro and type definitions ----------------------*/
/* SuperVGA information block */
struct
{
char VESASignature[4]; /* 'VESA' 4 byte signature */
short VESAVersion; /* VBE version number */
char far *OEMStringPtr; /* Pointer to OEM string */
long Capabilities; /* Capabilities of video card */
unsigned far *VideoModePtr; /* Pointer to supported modes */
short TotalMemory; /* Number of 64kb memory blocks */
char reserved[236]; /* Pad to 256 byte block size */
} VbeInfoBlock;
/* SuperVGA mode information block */
struct
{
unsigned short ModeAttributes; /* Mode attributes */
unsigned char WinAAttributes; /* Window A attributes */
unsigned char WinBAttributes; /* Window B attributes */
unsigned short WinGranularity; /* Window granularity in k */
unsigned short WinSize; /* Window size in k */
unsigned short WinASegment; /* Window A segment */
unsigned short WinBSegment; /* Window B segment */
void (far *WinFuncPtr)(void); /* Pointer to window function */
unsigned short BytesPerScanLine; /* Bytes per scanline */
unsigned short XResolution; /* Horizontal resolution */
unsigned short YResolution; /* Vertical resolution */
unsigned char XCharSize; /* Character cell width */
unsigned char YCharSize; /* Character cell height */
unsigned char NumberOfPlanes; /* Number of memory planes */
unsigned char BitsPerPixel; /* Bits per pixel */
unsigned char NumberOfBanks; /* Number of CGA style banks */
unsigned char MemoryModel; /* Memory model type */
unsigned char BankSize; /* Size of CGA style banks */
unsigned char NumberOfImagePages; /* Number of images pages */
unsigned char res1; /* Reserved */
unsigned char RedMaskSize; /* Size of direct color red mask */
unsigned char RedFieldPosition; /* Bit posn of lsb of red mask */
unsigned char GreenMaskSize; /* Size of direct color green mask */
unsigned char GreenFieldPosition; /* Bit posn of lsb of green mask */
unsigned char BlueMaskSize; /* Size of direct color blue mask */
unsigned char BlueFieldPosition; /* Bit posn of lsb of blue mask */
```

```
unsigned char RsvdMaskSize; /* Size of direct color res mask */
unsigned char RsvdFieldPosition; /* Bit posn of lsb of res mask */
unsigned char DirectColorModeInfo; /* Direct color mode attributes */
unsigned char res2[216]; /* Pad to 256 byte block size */
} ModeInfoBlock;

typedef enum
{
memPL = 3, /* Planar memory model */
memPK = 4, /* Packed pixel memory model */
memRGB = 6, /* Direct color RGB memory model */
memYUV = 7, /* Direct color YUV memory model */
} memModels;
/*--------------------------- Global Variables ----------------------------*/
char mystr[256];
char *get_str();
int xres,yres; /* Resolution of video mode used */
int bytesperline; /* Logical CRT scanline length */
int curBank; /* Current read/write bank */
unsigned int bankShift; /* Bank granularity adjust factor */
int oldMode; /* Old video mode number */
char far *screenPtr; /* Pointer to start of video memory */
void (far *bankSwitch)(void); /* Direct bank switching function */
/*------------------------ VBE Interface Functions ------------------------*/
/* Get SuperVGA information, returning true if VBE found */
int getVbeInfo()
{
union REGS in,out;
struct SREGS segs;
char far *VbeInfo = (char far *)&VbeInfoBlock;
in.x.ax = 0x4F00;
in.x.di = FP_OFF(VbeInfo);
segs.es = FP_SEG(VbeInfo);
int86x(0x10, &in, &out, &segs);
return (out.x.ax == 0x4F);
}
/* Get video mode information given a VBE mode number. We return 0 if
* if the mode is not available, or if it is not a 256 color packed
* pixel mode.
*/
int getModeInfo(int mode)
{
```

```
union REGS in,out;
struct SREGS segs;
char far *modeInfo = (char far *)&ModeInfoBlock;
if (mode < 0x100) return 0; /* Ignore non-VBE modes */
in.x.ax = 0x4F01;
in.x.cx = mode;
in.x.di = FP_OFF(modeInfo);
segs.es = FP_SEG(modeInfo);
int86x(0x10, &in, &out, &segs);
if (out.x.ax != 0x4F) return 0;
if ((ModeInfoBlock.ModeAttributes & 0x1)
&& ModeInfoBlock.MemoryModel == memRGB
&& ModeInfoBlock.BitsPerPixel == 32
/*&& ModeInfoBlock.NumberOfPlanes == 1*/)
return 1;
return 0;
}
/* Set a VBE video mode */
void setVBEMode(int mode)
{
union REGS in,out;
in.x.ax = 0x4F02; in.x.bx = mode;
int86(0x10,&in,&out);
}
/* Return the current VBE video mode */
int getVBEMode(void)
{
union REGS in,out;
in.x.ax = 0x4F03;
int86(0x10,&in,&out);
return out.x.bx;
}

/* Set new read/write bank. We must set both Window A and Window B, as
* many VBE's have these set as separately available read and write
* windows. We also use a simple (but very effective) optimization of
* checking if the requested bank is currently active.
*/
void setBank(int bank)
{
union REGS in,out;
if (bank == curBank) return; /* Bank is already active */
```

```
curBank = bank; /* Save current bank number */
bank <<= bankShift; /* Adjust to window granularity */
#ifdef DIRECT_BANKING
setbxdx(0,bank);
bankSwitch();
setbxdx(1,bank);
bankSwitch();
#else
in.x.ax = 0x4F05; in.x.bx = 0; in.x.dx = bank;
int86(0x10, &in, &out);
in.x.ax = 0x4F05; in.x.bx = 1; in.x.dx = bank;
int86(0x10, &in, &out);
#endif
}
/*------------------------ Application Functions -----------------------*/
/* Plot a pixel at location (x,y) in specified color (8 bit modes only) */
void putPixel(int x,int y,char color[3])
{

 long addr = (long)(y) * bytesperline + (x*4);

setBank((int)(addr >> 16));

*(screenPtr + (addr & 0xFFFF))  = color[0];
*(screenPtr + (addr & 0xFFFF)+1)  = color[1];
*(screenPtr + (addr & 0xFFFF)+2)  = color[2];
*(screenPtr + (addr & 0xFFFF)+3)  = 0xff;

}
/* Draw a line from (x1,y1) to (x2,y2) in specified color */
void line(int x1,int y1,int x2,int y2,int color)
{
int d; /* Decision variable */
int dx,dy; /* Dx and Dy values for the line */
int Eincr,NEincr; /* Decision variable increments */
int yincr; /* Increment for y values */
int t; /* Counters etc. */
#define ABS(a) ((a) >= 0 ? (a) : -(a))
dx = ABS(x2 - x1);
dy = ABS(y2 - y1);
if (dy <= dx)
{
```

```
/* We have a line with a slope between -1 and 1
*
* Ensure that we are always scan converting the line from left to
* right to ensure that we produce the same line from P1 to P0 as the
* line from P0 to P1.
*/
if (x2 < x1)
{
t = x2; x2 = x1; x1 = t; /* Swap X coordinates */
t = y2; y2 = y1; y1 = t; /* Swap Y coordinates */
}
if (y2 > y1)
yincr = 1;
else
yincr = -1;
d = 2*dy - dx; /* Initial decision variable value */
Eincr = 2*dy; /* Increment to move to E pixel */
NEincr = 2*(dy - dx); /* Increment to move to NE pixel */
putPixel(x1,y1,color); /* Draw the first point at (x1,y1) */
/* Incrementally determine the positions of the remaining pixels */
for (x1++; x1 <= x2; x1++)
{
if (d < 0)
d += Eincr; /* Choose the Eastern Pixel */
else
{
d += NEincr; /* Choose the North Eastern Pixel */
y1 += yincr; /* (or SE pixel for dx/dy < 0!) */
}
putPixel(x1,y1,color); /* Draw the point */
}
}
else
{
/* We have a line with a slope between -1 and 1 (ie: includes
* vertical lines). We must swap our x and y coordinates for this.
*
* Ensure that we are always scan converting the line from left to
* right to ensure that we produce the same line from P1 to P0 as the
* line from P0 to P1.
*/
if (y2 < y1)
```

```
{
t = x2; x2 = x1; x1 = t; /* Swap X coordinates */
t = y2; y2 = y1; y1 = t; /* Swap Y coordinates */
}
if (x2 > x1)
yincr = 1;
else
yincr = -1;
d = 2*dx - dy; /* Initial decision variable value */
Eincr = 2*dx; /* Increment to move to E pixel */
NEincr = 2*(dx - dy); /* Increment to move to NE pixel */
putPixel(x1,y1,color); /* Draw the first point at (x1,y1) */
/* Incrementally determine the positions of the remaining pixels */
for (y1++; y1 <= y2; y1++)
{
if (d < 0)
d += Eincr; /* Choose the Eastern Pixel */
else
{
d += NEincr; /* Choose the North Eastern Pixel */
x1 += yincr; /* (or SE pixel for dx/dy < 0!) */
}
putPixel(x1,y1,color); /* Draw the point */
}
}
}
/* Draw a simple moire pattern of lines on the display */
void drawMoire(void)
{
int i,j,x;
int v;
char mystr0[3];
	FILE *fp;
	fp = fopen("fady.bmp","rb");
	for(x = 1 ; x <= 54 ; x++ )
		mystr0[0] = fgetc(fp);

for (i = 0; i < yres; i++)
{
	for (j = 0 ; j < xres; j++)
	{
		mystr0[0] = fgetc(fp); /* b */
```

```
            mystr0[1] = fgetc(fp);  /* g */
            mystr0[2] = fgetc(fp);  /* r */
        putpixel(j,yres-i+1,mystr0);
        }
}
    fclose(fp);
}

/* Return NEAR pointer to FAR string pointer*/
char *get_str(char far *p)
{
int i;
char *q=mystr;
for(i=0;i<255;i++)
{
if(*p) *q++ = *p++;
else break;
}
*q = '\0';
return(mystr);
}
/* Display a list of available resolutions. Be careful with calls to
* function 00h to get SuperVGA mode information. Many VBE's build the
* list of video modes directly in this information block, so if you
* are using a common buffer (which we aren't here, but in protected
* mode you will), then you will need to make a local copy of this list
* of available modes.
*/
void availableModes(void)
{
unsigned far *p;
if (!getVbeInfo())
{
printf("No VESA VBE detected\n");
exit(1);
}
printf("VESA VBE Version %d.%d detected (%s)\n\n",
VbeInfoBlock.VESAVersion >> 8, VbeInfoBlock.VESAVersion & 0xF,
get_str(VbeInfoBlock.OEMStringPtr));
printf("Available 256 color video modes:\n");
for (p = VbeInfoBlock.VideoModePtr; *p !=(unsigned)-1; p++)
{
```

```
if (getModeInfo(*p))
{
printf(" %4d x %4d %d bits per pixel\n",
ModeInfoBlock.XResolution, ModeInfoBlock.YResolution,
ModeInfoBlock.BitsPerPixel);
}
}
printf("\nUsage: hellovbe <xres> <yres>\n");
exit(1);
}
/* Initialize the specified video mode. Notice how we determine a shift
* factor for adjusting the Window granularity for bank switching. This
* is much faster than doing it with a multiply (especially with direct
* banking enabled).
*/
void initGraphics(unsigned int x, unsigned int y)
{
unsigned far *p;
if (!getVbeInfo())
{
printf("No VESA VBE detected\n");
exit(1);
}
for (p = VbeInfoBlock.VideoModePtr; *p != (unsigned)-1; p++)
{
if (getModeInfo(*p) && ModeInfoBlock.XResolution == x
&& ModeInfoBlock.YResolution == y)
{
xres = x; yres = y;
bytesperline = ModeInfoBlock.BytesPerScanLine;
bankShift = 0;
while ((unsigned)(64 >> bankShift) != ModeInfoBlock.WinGranularity)
bankShift++;
bankSwitch = ModeInfoBlock.WinFuncPtr;
curBank = -1;
screenPtr = (char far *)( ((long)0xA000)<<16 | 0);
oldMode = getVBEMode();
setVBEMode(*p);
return;
}
}
printf("Valid video mode not found\n");
```

```
exit(1);
}
/* Main routine. Expects the x & y resolution of the desired video mode
* to be passed on the command line. Will print out a list of available
* video modes if no command line is present.
*/
void main(int argc,char *argv[])
{
initGraphics(800,600); /* Start requested video mode */
drawMoire(); /* Draw a moire pattern */
getch(); /* Wait for keypress */
setVBEMode(oldMode); /* Restore previous mode */
}
```

ويستخدم البرنامج Module بلغة التجميع .Asm

```
public _setbxdx
.MODEL SMALL ;whatever
.CODE
set_struc struc
dw ? ;old bp
dd ? ;return addr (always far call)
p_bx dw ? ;reg bx value
p_dx dw ? ;reg dx value
set_struc ends
_setbxdx proc far ; must be FAR
push bp
mov bp,sp
mov bx,[bp]+p_bx
mov dx,[bp]+p_dx
pop bp
ret
_setbxdx endp
END
```

تم ترجمة البرنامج واختباره باستخدام Microsoft C 800c Compiler والذى ياتى مع Visual C/C++ 1.55 وقد تم ترجمة برنامج الاسمبلى الصغير باستخدام MASM 6.1. وكان الملف الناتج عبارة عن ملف EXE ضغير الحجم مساحته 10,495 byte يعمل تحت نظام DOS.



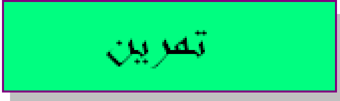
شكل(٧) – صورة TRUE COLOR تم عرضها تحت نظام DOS من خلال برنامج بلغة سى يستخدم VESA 3.0

ملحوظة هامة

للحصول على المزيد من المعلومات التى تتعلق بـ VESA

World Wide Web: www.vesa.org
E-mail: support@vesa.org
Fax: 408-957-9277
Voice: 408-957-9270

Mail to:
Video Electronics Standards Association
920 Hillview Court, Suite 140
Milpitas, CA 95035



بادر الان بمحاولة كتابة واجهة النظام الخاصة بك – من خلال هذا المستوى –برمجة العتاد Low Level Programming
كم تتخيل – ماهى المدة الازمة للنجاح فى هذه المهمة ؟ يوم – شهر – سنة ؟ ام اكثر !

المستوى الثانى لبرمجة واجهة النظام Graphic Library:-

كما هو معلوم ان علم هندسة البرمجيات علم تراكمى – وان بناء البرمجيات اشبه كثيرا ببناء ناطحات السحاب – الامر الذى يترتب عليه بناء الادوار السفلية – ثم الادوار العلوية – وهكذا

حتى لا يعانى المبرمجين من برمجة العتاد وحفظا للوقت وتوفيرا للجهد – تم الانتقال الى المرحلة الثانية من برمجة واجهة النظم – وهذه المرحلة هى توفير مكتبات للعرض فى حالة النمط النصى Text Mode او مكتبات للرسم فى حالة النمط الرسومى Graphic Mode – وعندها ينبغى فقط للميرمج ان يدرك كيفية استخدام هذه المكتبات وان لا يهتم بالعتاد على الاطلاق.

اذا كنت ترغب فى تطوير مكتبة جرافك خاصة بك – فانت بحاجة الى العمل فى المستوى السابق اى برمجة العتاد – الامر الذى يعنى الكثير من الوقت والمجهود – ولكن ذلك لا ينفى ضرورة مثل هذا العمل فى بعض الاحيان.

س : ماهى الوظائف الازمة فى مكتبة الجرافك ؟
جـ : هذا امر يحدده الغرض الذى سوف تستخدم فى المكتبة (معالجة الصور – الالعاب – واجهة التطبيقات – غير ذلك) ومع
ذلك سوف ندرس الملامح الاساسية لمكتبات الجرافك.

الملامح الاساسية لمكتبات العرض او الجرافك :-

- التعرف على الشاشة
- اختيار نمط الشاشة Screen Mode والذى يتحدد بالـ Resolution & COLORS
- الاشكال الاساسة للرسم (النقط - الخطوط – المربعات – الدوائر - ....وغيرها)
- التعامل مع ملفات الخطوط Fonts
- التعامل مع ملفات الصور Bitmap (فى حالة الجرافك)
- مناطق القص Clipping والاستبعاد Excluding
- الطبقات Layers (ميزة اضافية – لانه تدخل ضمن واجهة التفاعل (GUI
- دعم الفارة Mouse (ميزة اضافية)
- التعامل مع ملفات الفيديو (ميزة اضافية – لانه يدخل ضمن تعدد الوسائط)
- التعامل مع ملفات الصوت (ميزة اضافية – لانه يدخل ضمن تعدد الوسائط)

وفى الواقع يوجد العديد – بل المئات من مكتبات الجرافك – المخصصة لانواع مختلفة من نظم التشغيل – لغات البرمجة وتتنوع اغراض تلك المكتبات.

من امثلة مكتبات الجرافك المكتبة Allegro المخصصة للالعاب (بلغة سى) وتتدعم النظم التالية

DOS/djgpp
DOS/Watcom
Windows/MSVC
Windows/MinGW32

Windows/Cygwin
Windows/Borland
Linux (console)
Unix (X)
BeOS
QNX
MacOS/MPW

ويمكن الحصول على المكتبة من الموقع http://www.allegro.cc
وقد ساهم فى هذه المكتبة العديد من المبرمجين المبدعين – وعلى راسهم قائد المجموعة المبرمج Shawn Hargreaves

وايضا من مكتبات الجرافك – المكتبة GRX 2.4.6 والتى تدعم النظم DOS, Linux, X11 and Win32 وهى من ابداع المبرمج Csaba Biegl الذى قام ببرمجتها عام ١٩٩٢ وطورها Mariano Alvarez Fernández من عام ٢٠٠٠ حتى ٢٠٠٣

وحيث ان زمن مكتبات الجرافك يذكرنا كثيرا بعالم برمجة DOS ذلك النظام القديم – وحيث ان اللغة الاكثر شيوعا تحت هذا النظام هى لغة البرمجة الشهيرة CA-Clipper وهى لغة برمجة متخصصة فى انظمة قواعد البيانات وتم تطويرها على هذا الاساس فى البداية باستخدام لغة سى – ثم بعد ذلك تحولت الى لغة برمجة عامة لتطوير مختلف انواع التطبيقات مثل لغة سى التى بدات كلغة متخصصة فى برمجة نظم التشغيل ثم تحولت الى لغة برمجة عامة.

يوجد العديد من مكتبات الجرافك للغة كليبر CA-Clipper والتى تم تطويرها لها باستخدام لغة سى – مثل

1 – Light Lib Graphic Library
2 – DGE5
3 – FGLIB 3.1

وهذه المكتبات تتميز بالقوة والمرونة – ومن اهمها المكتبة Light Lib لانها باتت يتم توزيعها مع اللغة (الاصدار 5.3 من كليبر) يليها فى الاهمية المكتبة FGLIB 3.1 لانها كانت مجانية و لانه يتم تطويرها حتى الان (منذ عام ١٩٩٣ حتى عام ٢٠٠٦) حتى بعد انتهاء زمن DOS وزمن Clipper لانه يوجد العديد بل الالاف من البرامج التى تم تطويرها باللغة وبحاجة الى تحديث.

ادركت الشركات المنتجة للغات البرمجة مدى اهمية وجود مكتبة جرافك – داخل اللغة ولهذا نجد العديد من دوال الرسم و الجرافك داخل لغات البرمجة مثل Borland C/C++ وكذلك Microsoft C/C++ وغيرها.

قد تجد لغات برمجة متخصصة فى برمجة الالعاب مثل Euphoria 2.5 والتى تدعم الجرافك بقوة.

سوف ندرس الان مثال على كيفية استخدام مكتبة جرافك – هذا المثال هو المكتبة FGLIB 3.1 المخصصة للغة البرمجة

5.2e CA-Clipper ويمكـــن الحصـــول علـــى المكتبـــة مـــن الـــرابط http://www.sourceforge.net/projects/fglib وذلك من خلال

تحميل الملف FGLIB31.ZIP وحيث ان هذا الاصدار لا يشتمل على Help لانه كان اصدار تطوير بسيط من Vesa 1.0 الى Vesa 2.0 و Vesa 3.0 فيمكنك الحصول على الاصدار FGLib 3.0 من الموقع http://www.the-oasis.net والجدير بالذكر ان هذا الموقع هو اكبر موقع للشفيرات المصدرية والمكتبات Libraries الخاصة باللغة Clipper هذا بالنسبة للمهتمين بتلك اللغة ومن كانت لديهم تطبيقات قديمة سبق تطويرها بهذه اللغة العتيقة (بدات عام ١٩٨٤ وانتهت عام ١٩٩٧)

مدخل الى المكتبة FGLIB :-

سوف نتعرف الان على كيفية استخدام المكتبة FGLIB من خلال لغة Clipper بداية يتم استدعاء ملف Header (لا يشتمل على prototype للدوال كما هو الحال فى لغة سى) والذى يشتمل على تعريفات لمتغيرات يسهل استخدامها التعامل مع المكتبة.
يلى ذلك استدعاء دوال من المكتبة

```
#include "FGL.CH"
*......... Call  Library functions
```

ان المكتبة تم بنائها على البرمجة الهيكلية Programming Structure بلغة سى وليس سى بلس بلس ++C كما ان لغة البرمجة كليبر مبنية على البرمجة الهيكلية Structure Programming ايضا (ولكن يوجد مكتبات لها مثل Class(Y) والتى تضيف نمط برمجة الكائنات الى اللغة – ويوجد المكتبة DoubleS والتى تضيف نمط برمجة الخادم الممتاز الى اللغة).

وبعد كتابة البرنامج وحفظه فى ملف PRG. (بدل من C. كما فى لغة سى او CPP. كما فى سى بلس بلس) فانه يمكن ترجمة البرنامج كالتالى

```
Clipper myfile.prg
```

وعندها نحصل على ملف Object بالامتداد myfile.obj والان ياتى دور عملية الربط Linking لاستخراج ملف جاهز للتنفيذ يحمل الامتداد EXE. (لا تستطيع اللغة كليبر استخراج ملفات COM. كما هو الحالى فى لغة سى واسمبلى)

```
RTLINK FI myfile LIB FGLIB31
```

حيث FI هى اختصار لـ FILE وهى جزء من الـ Syntax الخاص بالرابط RTLINK وكما هو واضح تم الاشارة الى المكتبة FGLIB31.LIB والتى تحمل دوال الجرافكس التى قمنا باستخدامها.

ملحوظة هامة

يوجد العديد من برامج الربط التى يمكن استخدامها مثل :-

1 – Microsoft Link
2 – Blinker
3 – Rtlink
4 – Causeway
5 - Exospace

وقد تم اختبار كل من RTLINK و Blinker وهما يعملان بصورة جيدة – الافضل هو Blinker لانه يدعم Protected Mode
ويستخدم بصورة مشابهة لـ RTLINK كالتالى

```
Blinker FI myfile LIB FGLIB31
```

ايضا يمكن استخدام ملفات LNK. يوضح فيها المعلومات الازمة لاجراء عملية الربط – ثم يتم استدعائها حتى تتم عملية
الربط مباشرة – كالتالى Blinker @mylink حيث mylink.LNK هو الملف الذى يشمل المعلومات اللازمة للربط.
والان جاء دور كتابة برنامج يقوم بعمل شى

```
* MYTEST.PRG
 #include "FGL.CH"
FGLSetMode(FGL_GRAPHICS_640_480_16)
FGLFillRectangle(0,0,640,480, clBlue )
INKEY(0)
FGLSetMode(3)
```

هذا البرنامج البسيط يقوم بالدخول على النمط الرسومى للشاشة 640 x 480 x 16 colors ثم يلون الشاشة باللون الازرق
ثم يقوم بعد ذلك بالانتظار حتى يضغط المستخدم اى مفتاح – ثم يعود للنمط النصى
نلاحظ ان الدالة ()FGLSETMODE تستخدم من اجل اختيار نمط الشاشة Screen Mode والذى يتم تحديده برقم معين او من خلال متغير او ثابت معرف سابقا كبديل لادخال الرقم الدالة ()FGLFillrectangle تستخدم لمسح مساحة مستطيلة من الشاشة بلون محدد.
يمكن من خلال التعليمات الخاصة بالمكتبة التعرف عى جميع الدوال المتاحة بها وكيفية استخدامها – لكن سوف نعرض الان مفهومين ومصطلحين فى غاية الاهمية

VESA :- هى اختصار لـ Video Electronics Standards Association
وعندما يقوم كارت الشاشة بدعمها فانه يمكن برمجته بطريقة قياسية. وهناك اصدارات مختلفة من VESA مثل VESA 1.0 , VESA 1.2 , VESA 2.0 & VESA 3.0

SCREEN MODE :- نمط الشاشة والذى يعرف بالعرض والارتفاع وعدد الالوان Width,Height & Colors
والالوان هى نقطة الحوار وبناء عليها يتحدد مدى جودة عرض الصور.
عدد الالوان يتم تحديده من خلال عدد BITS المستخدمة فى تمثيل اللون.

- فمثلا 1 BIT تعنى ان عدد الالوان المتاحة هو 2 Colors
- 4 Bits تعنى ان عدد الالوان المتاحة هو 16 colors

- 8 Bits تعنى ان عدد الالوان المتاحة هو 256 colors
- 15 Bits تعنى ان عدد الالوان المتاحة هو 32k colors
- 16 Bits تعنى ان عدد الالوان المتاحة هو 64k colors
- 24 bits تعنى ان عدد الالوان المتاحة هو 16M colors
- 32 bits تعنى ان عدد الالوان المتاحة ! يضا هو 16M colors

وبالتالى كلما زاد عدد الالوان كلما ازدات المساحة اللازمة لتخزين الصورة اما فى وحدات التخزين من الاقراص Disk Storage
Unit او من الذاكرة العشوائية RAM

ولعلك تسال لماذا نستخدم 32 Bits طالما ان عدد الالوان كما هو 16M ولم يزيد ؟ ببساطة ذلك يسهل التعامل مع الذاكرة لتمثيل النقطة الواحدة ONE PIXEL حيث نستخدم Double Word اى DWORD.

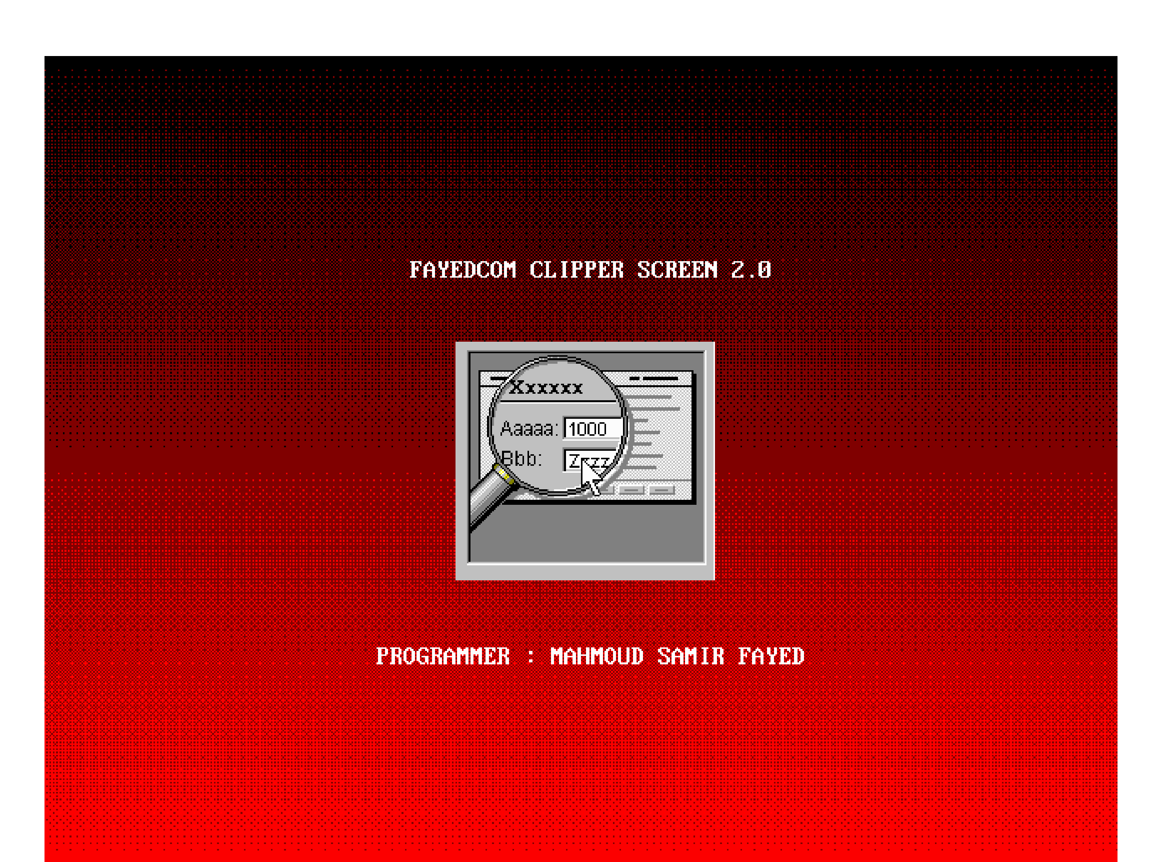
من العوامل المهة فى برمجة الجرافكس هى الذاكرة Memory لذلك ينصح بشدة ان تعمل برامج الجرافك فى Protected Mode حتى يتاح لها ذاكرة اكبر 16 Mega Byte (ذاكرة كبيرة بالنسبة للبرامج القديمة التى تعمل تحت DOS وصممت على العمل كـ 16Bit Applications).

ملحوظة هامة

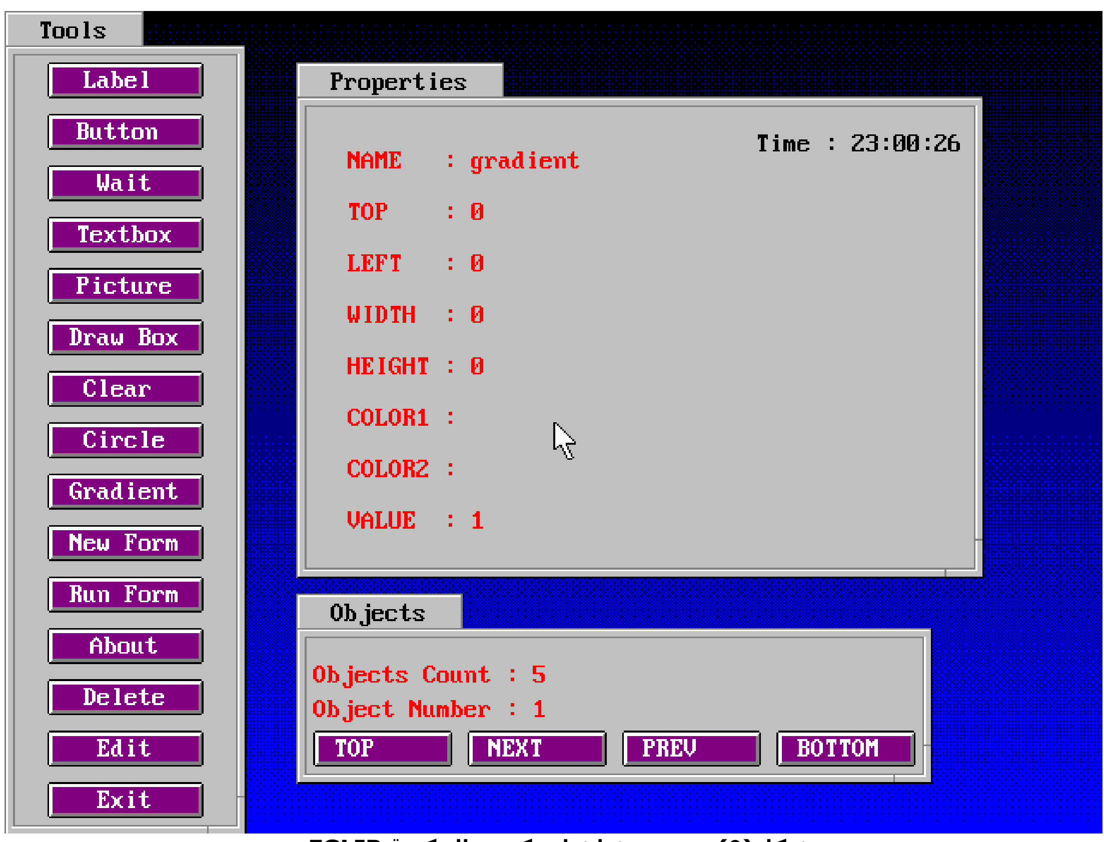
ان المكتبة FGLIB من المكتبات الغنية والتى تدعم انماط الشاشة المختلفة – بالاضافة الى دعم كامل للخطوط يتيح لك استخدام ملفات الخطوط FNT (اذا كنت تريد استخدام ملفات TTF فانه يوجد العديد من البرامج للتحويل من TTF الى FNT)
كما انها تدعم الصور بانواع مختلفة (BMP & PCX) وتم دعم GIF ايضا فى الاصدار FGLIB 3.1 كما يوجد العديد من الدوال التى تخدم اغراض الرسم Drawing مما يتيح رسم الاشكال المختلفة.
ان ملفات التعليمات الخاصة باستخدام المكتبة ثلاثة انواع هى TXT و DOC و NG لذلك يمكنك الاطلاع على التعليمات اما باستخدام Notepad او Microsoft Word او برنامج Norton Guide الخاص بملفات NG

س : ما هى امكانيات واجهة البرامج المصممة بمكتبة جرافك مثل FGLIB ؟

يتوقف ذلك حسب امكانيات مكتبة الجرافك – ومع ذلك طالما انها مكتبة جرافك فقط ومازلنا فى المستوى الثانى لبرمجة واجهة النظام فان الامكانيات المتاحة تعد بسيط جدا – والاشكال التالية تبين نتائج بسيطة تم الحصول عليها كواجهة للبرامج رغم بذل مجهود كبير – مما يعنى ان الوقوف عند مستوى مكتبة الجرافك امر غير مقبول – وينبغى الانتقال الى مستوى اخر يعطى ملامح اكبر لواجهة التطبيقات.



شكل (٨) – بداية برنامج كتب بالمكتبة FGLIB

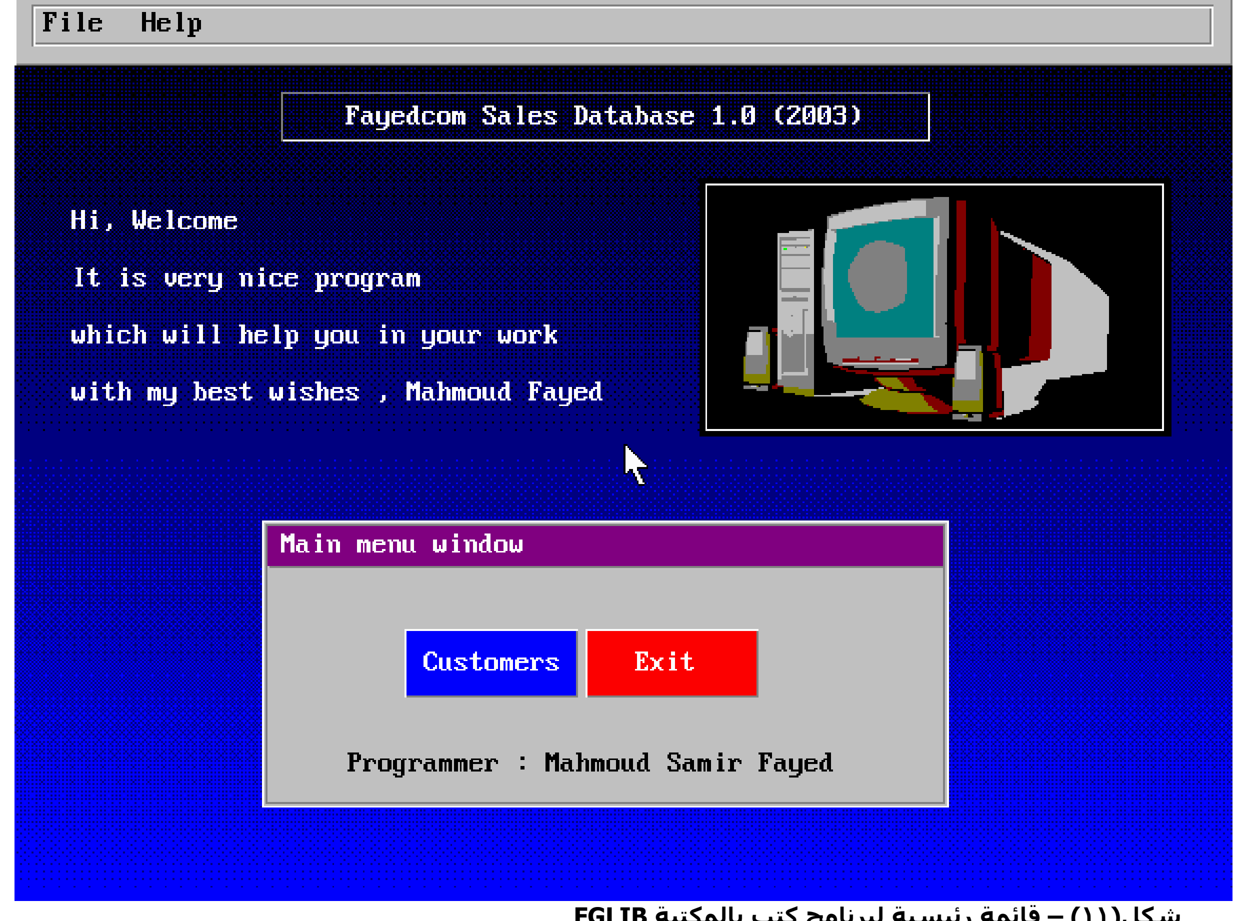


شكل(٩) مصمم شاشات كتب بالمكتبة FGLIB

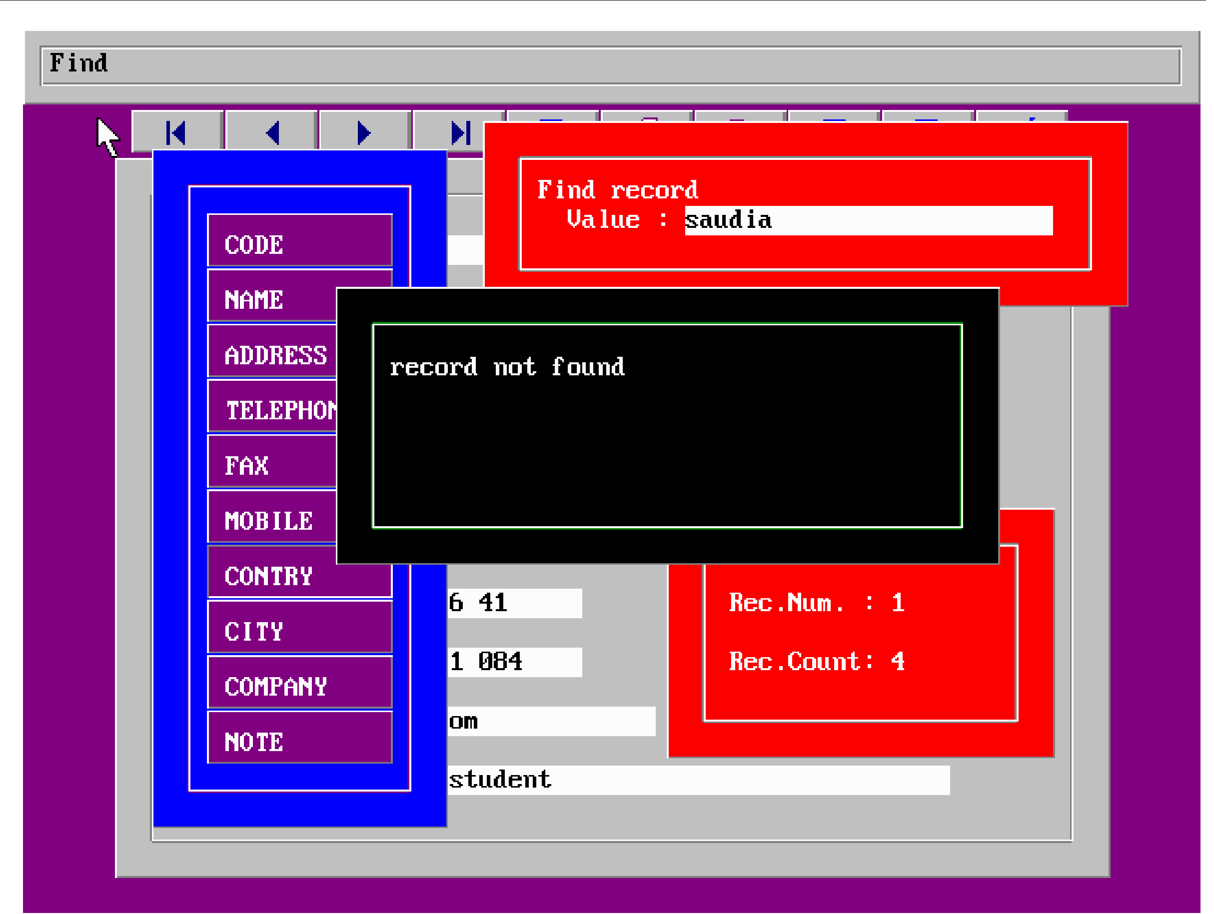
FayedCom

Loading....

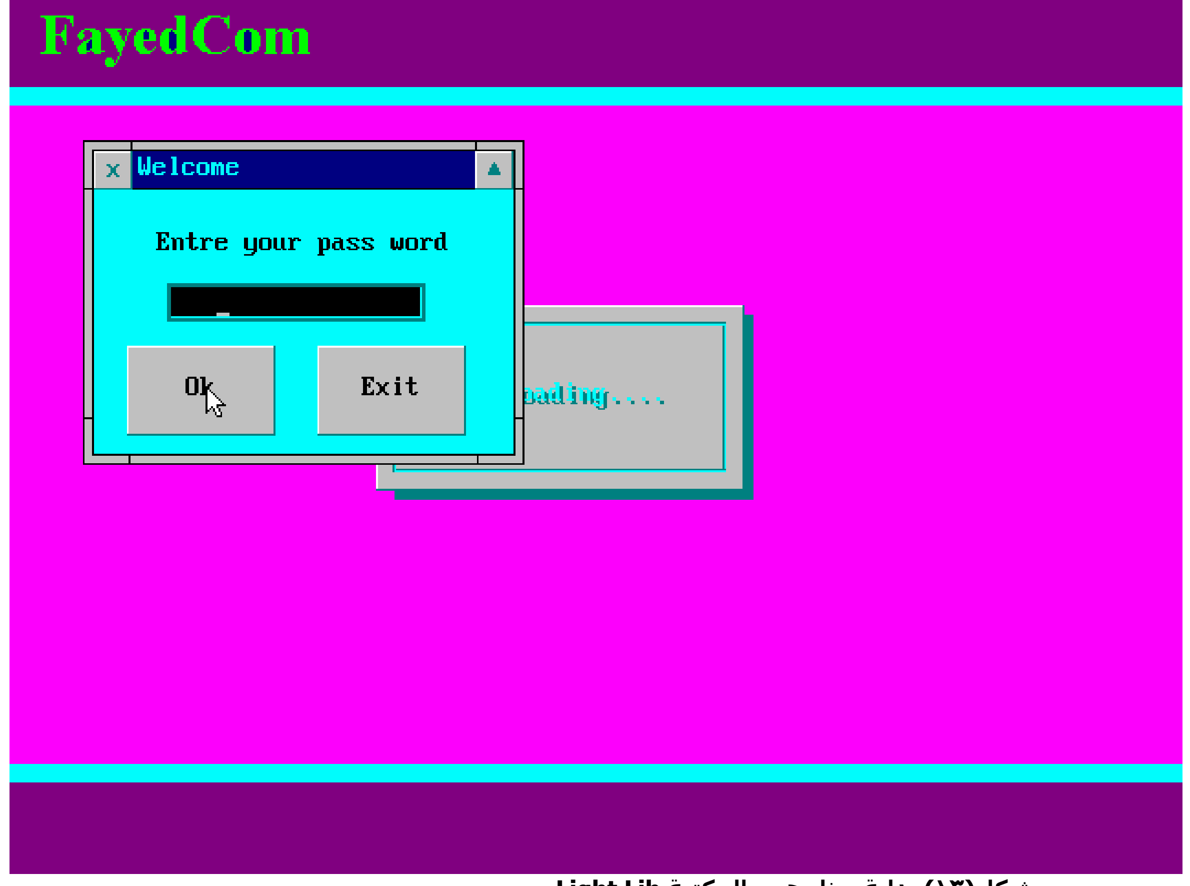
شكل(١٠) – بداية برنامج كتب بالمكتبة FGLIB



شكل(١١) – قائمة رئيسية لبرنامج كتب بالمكتبة FGLIB



شكل(١٢) – شاشة بيانات العملاء – بالمكتبة FGLIB



شكل(١٣) بداية برنامج – بالمكتبة Light Lib

Fayedcom Medical Database 1.0 (2004)

File Data System

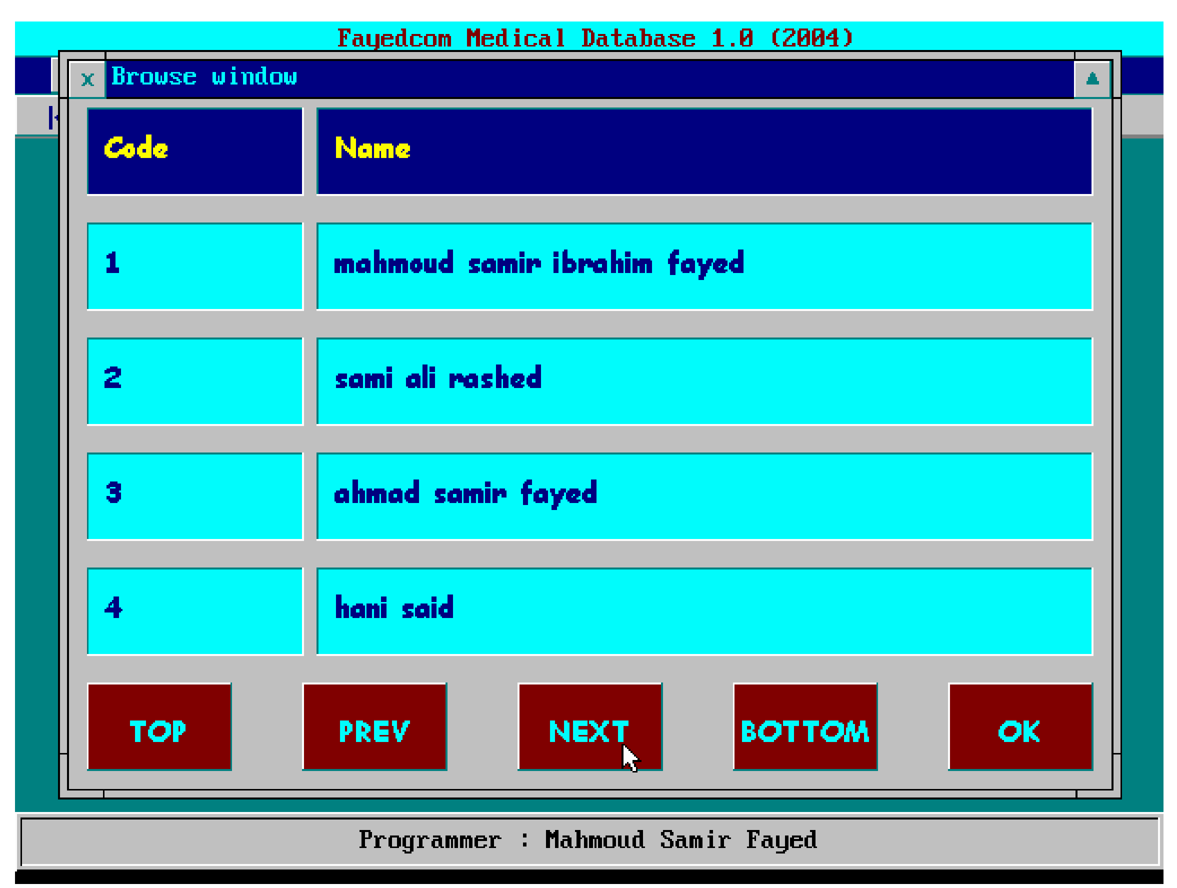
Medical Ver 1.0

Data Window

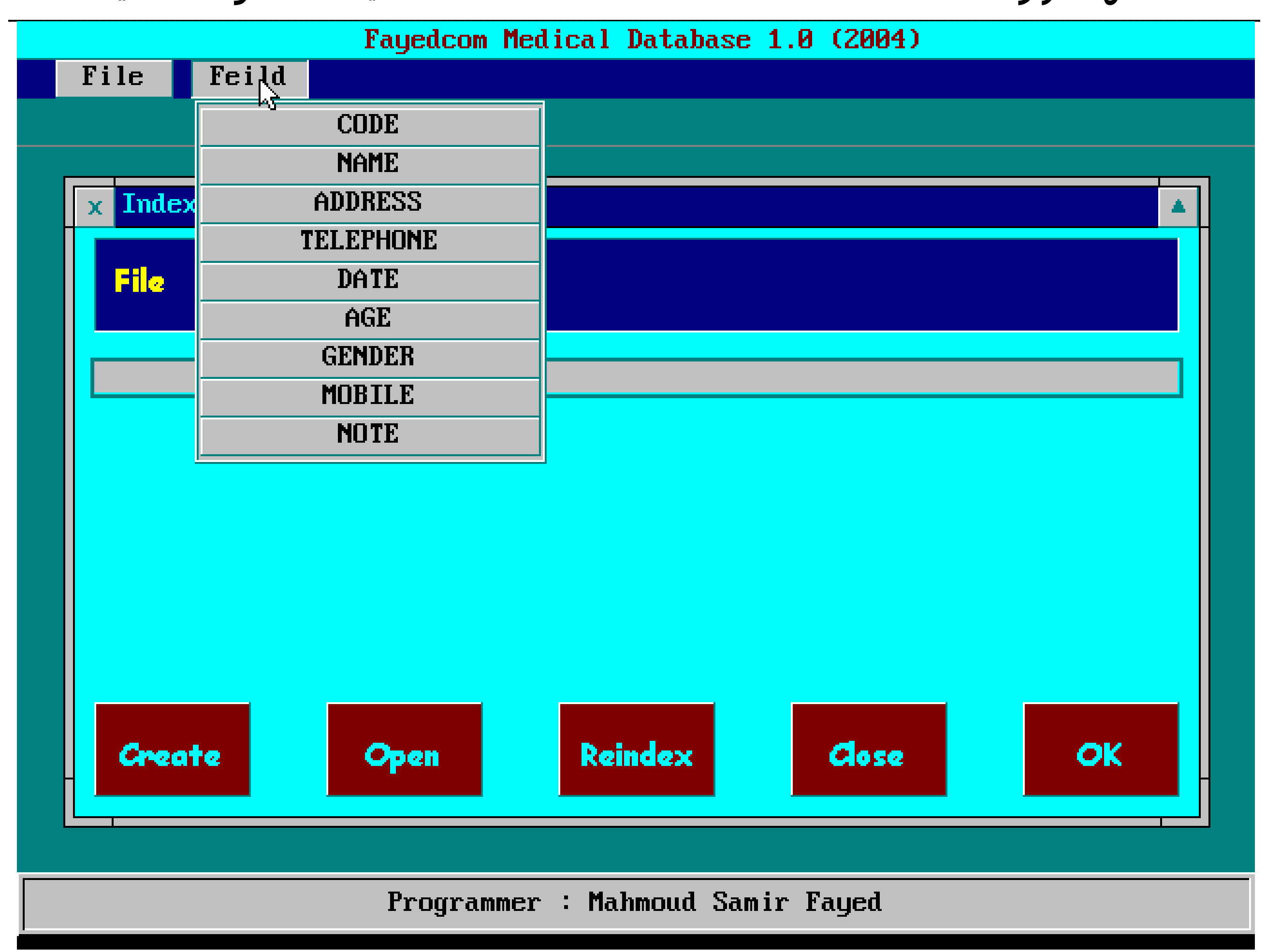
Code :	1
Name :	mahmoud samir ibrahim fayed
Gender :	male
Age :	17
Address :	jeddah
Telephone :	6295881
Mobile :	000
Date :	12\01\2000
Note :	he is eng.

Programmer : Mahmoud Samir Fayed

شكل(١٤) شاشة بيانات – بالمكتبة Light Lib



شكل(١٥) شاشة استعراض بيانات – المكتبة Light Lib



شكل(١٦) شاشة فهرسة بيانات – المكتبة Light Lib

ملحوظة هامة

ان برمجة واجهة النظام من خلال مكتبة جرافك فقط – امر مزعج جدا ويتطلب الكثير من الوقت والمجهود والصبر لكتابة الالاف من الاسطر البرمجية – هذا مما لايتحمله العديد بل الكثير من المبرمجين من اجل واجهة النظام – لهذا تجد ان فى تلك المرحلة كان المبرمجين يفضلون عمل تطبيقات تعمل فى Text Mode وان يتم التحويل الى Graphic Mode اثناء عمل البرنامج عند الضرورة فقط – كعرض صورة فى مقدمة البرنامج (Logo Screen) او لعرض الرسوم البيانية وهكذا.

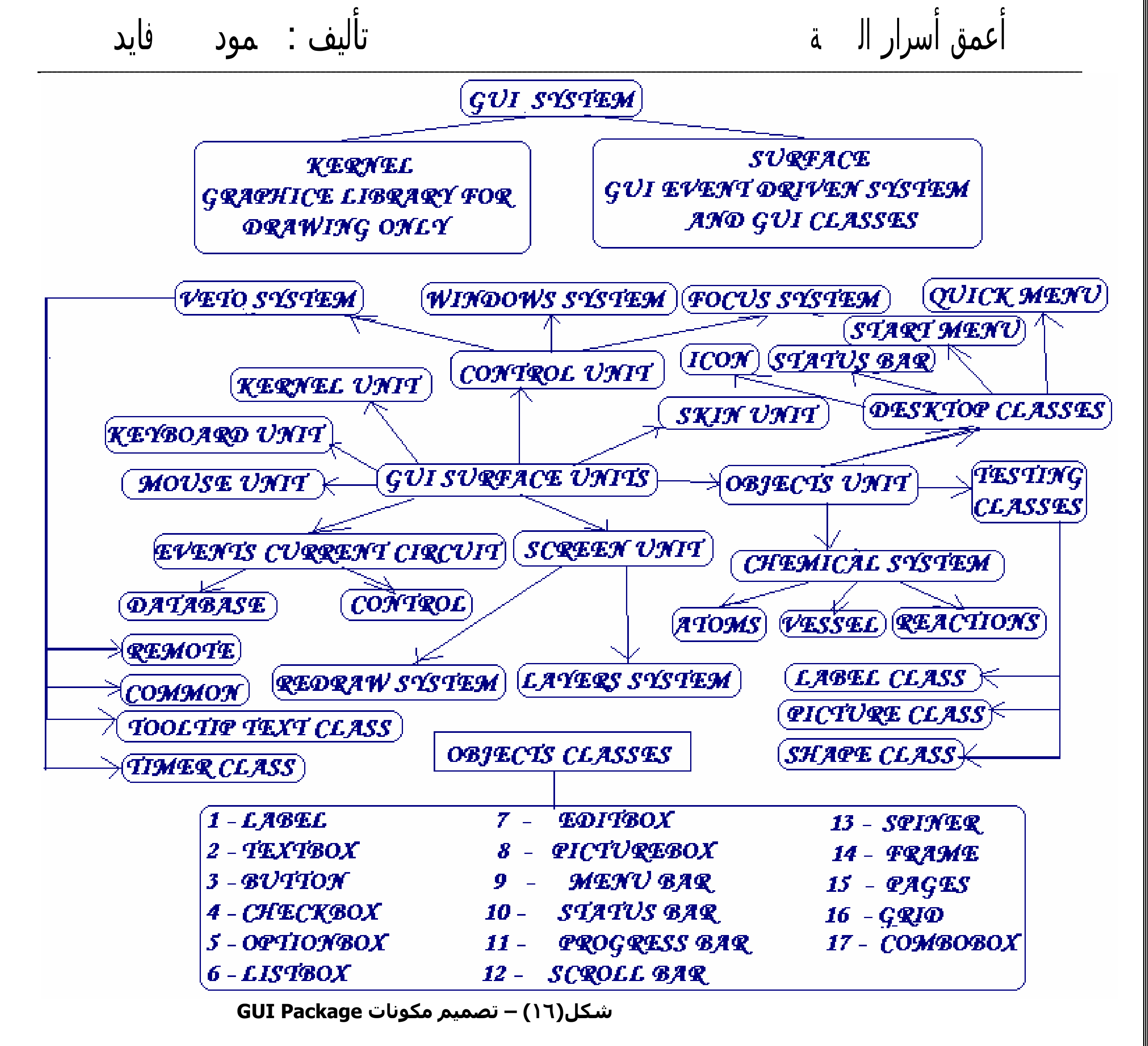
المستوى الثالث لبرمجة واجهة النظام GUI Package:-

فى هذا المستوى تتوفر لدينا الادوات الازمة لتطوير واجهة النظام (الجرافك + التفاعل مع المستخدم) بدون الحاجة الى تطوير نظام التفاعل مع المستخدم من البداية.
فى هذا المستوى توفر لنا GUI Package الخصائص التالية
١- نظام ادارة احداث مختبى داخل الواجهة
٢- نظام التفاعل مع لوحة المفاتيح والفارة

٣- نظام لادارة موارد الشاشة (اعادة رسم الشاشة Redraw System– الطبقات Layers System)
٤- طبقة ربط مع مكتبة الجرافك لاستخدام خصائصها – والنظم التى قام ببرمجتها محترفين تدعم اكثر من مكتبة جرافك
٥- نظام لادارة البورة Focus System اما من خلال الفارة او من خلال لوحة المفاتيح
٦- نظام لادارة النوافذ (تعدد النوافذ – نوافذ ديناميكية يمكن تحريكها وتحجيمها)
٧- عناصر التحكم المختلفة Controls او GUI Widgets مثل ازرار الاوامر ومربع النص وهكذا
   - Label
   - TextBox
   - EditBox
   - Command Button
   - ListBox
   - Compo Box
   - Shape
   - Image
   - Pages/Tabs
   - Tree
   - Grid
   - Scrool Bar
   - Frame
   - CheckBox
   - OptionBox
   - Timer
   - StatusBar
   - MenuBar
   - ToolBar

وغيرها من العناصر المختلفة التى تدعم تصميم واجهة النظام.

وقد نحتاج لتصميم GUI Package بانفسنا مستخدمين خبرتنا بالعمل فى المستوى السابق – وقد نستخدم Package جاهرة انظر شكل (١٦) والذى يوضح تصميم لمكونات GUI Package قام المولف بعملها منذ سنوات وبالتحديد عام 2005



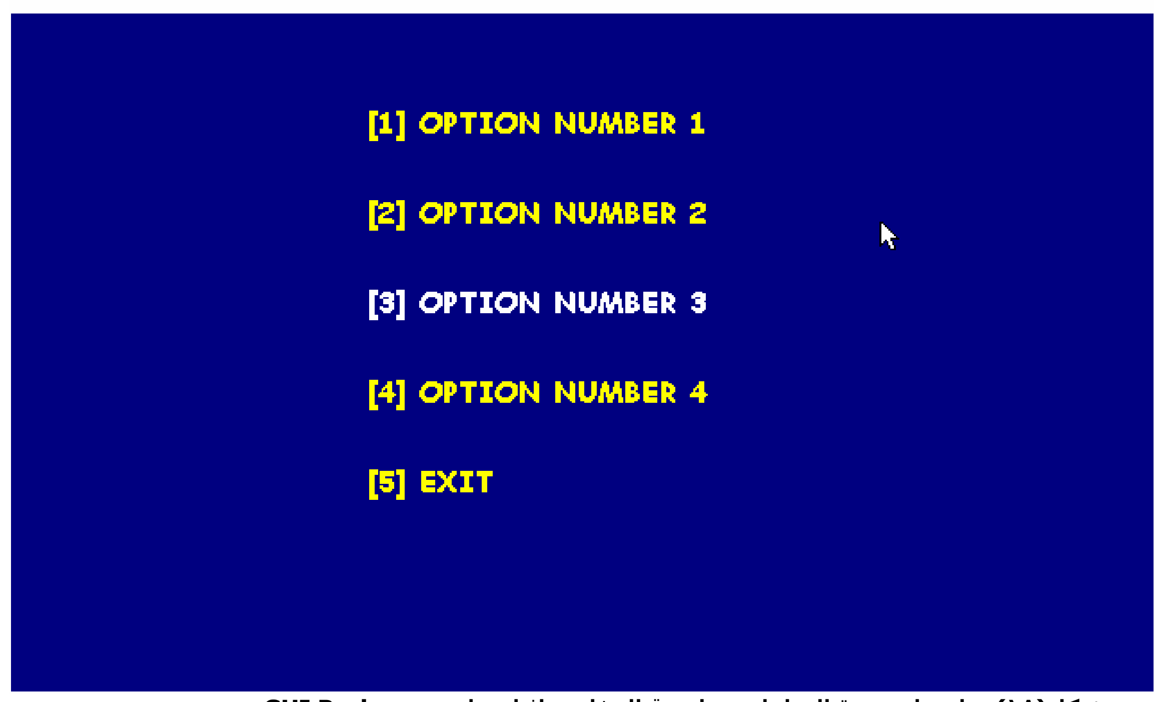
شكل(١٦) – تصميم مكونات GUI Package

والاشكال التالية من (١٧) حتى (٢٧) توضح العمل فى تطوير نظام GUI Package مما يعطى للقارى تصور للمجهود المبذول وراء تطوير GUI Package من البداية – مما يوضح مدى الفائدة من استخدام GUI Package جاهزة مما يوفر العديد من الوقت والمجهود.

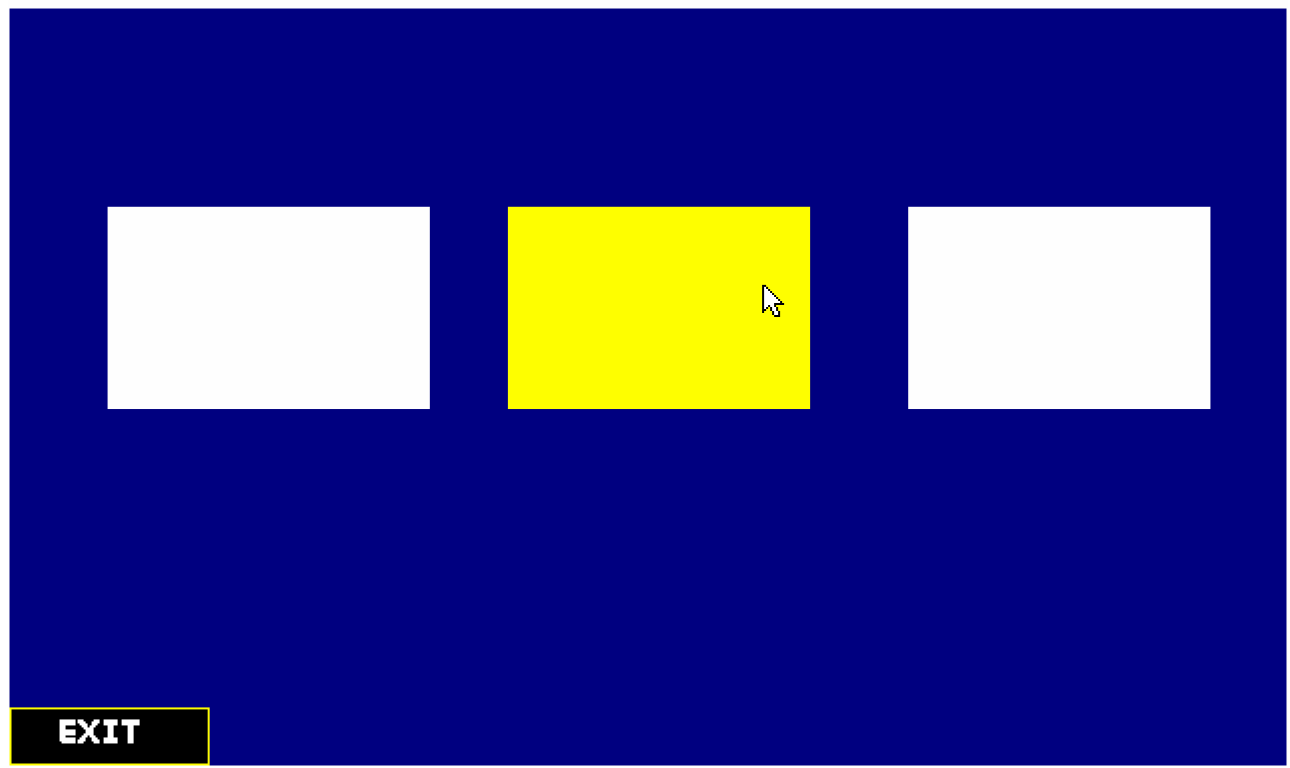
يمكن الحصول على الشفيرة المصدرية لهذا المشروع من الموقع
http://www.sourceforge.net/projects/fglib الملف FGLGUI3.ZIP وهو من عمل المولف استنادا على مكتبة الجرافك FGLib.

14
15
KERNEL TEST PROGRAM
305
60

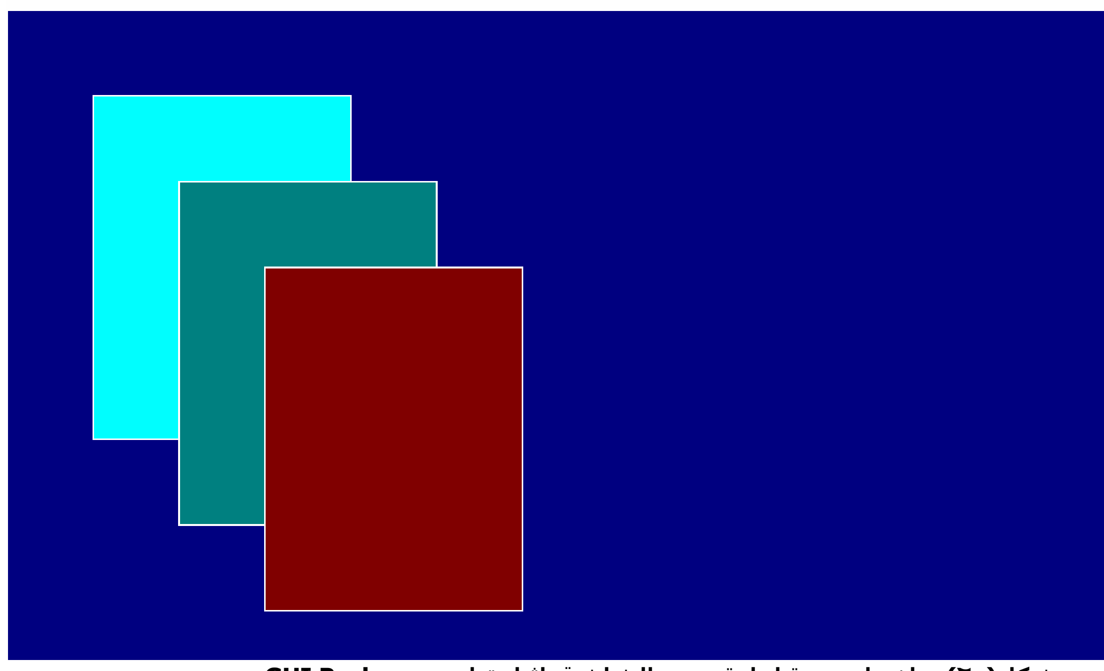
شكل(١٧) – اختبار الـ Kernel وهى عبارة عن ربط Interface بين GUI و Graphic Library



شكل(١٨) – اختبار وحدة التعامل مع لوحة المفاتيح اثناء تطوير GUI Package



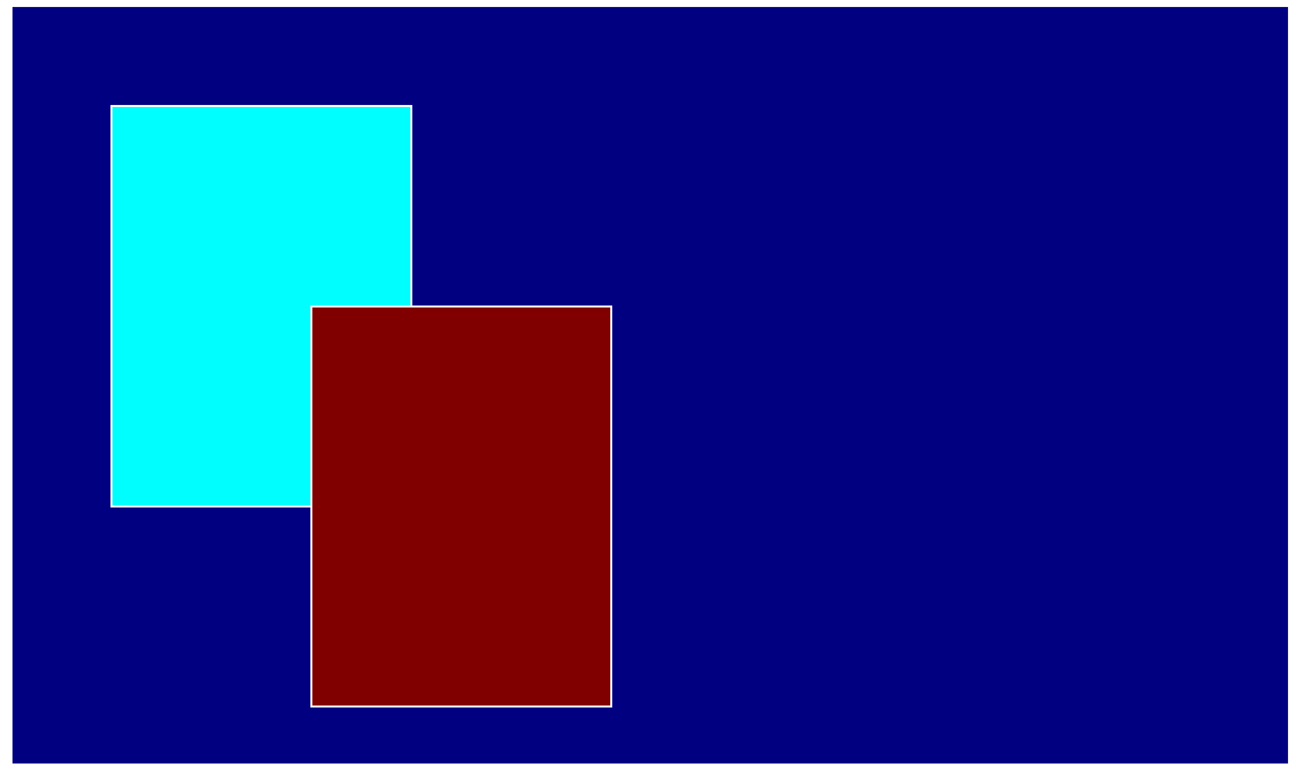
شكل(١٩) – اختبار وحدة التعامل مع الفارة اثناء تطوير GUI Package



شكل(٢٠) – اختبار وحدة إعادة رسم الشاشة اثناء تطوير GUI Package

SUPER GUI DESIGN PROJECT

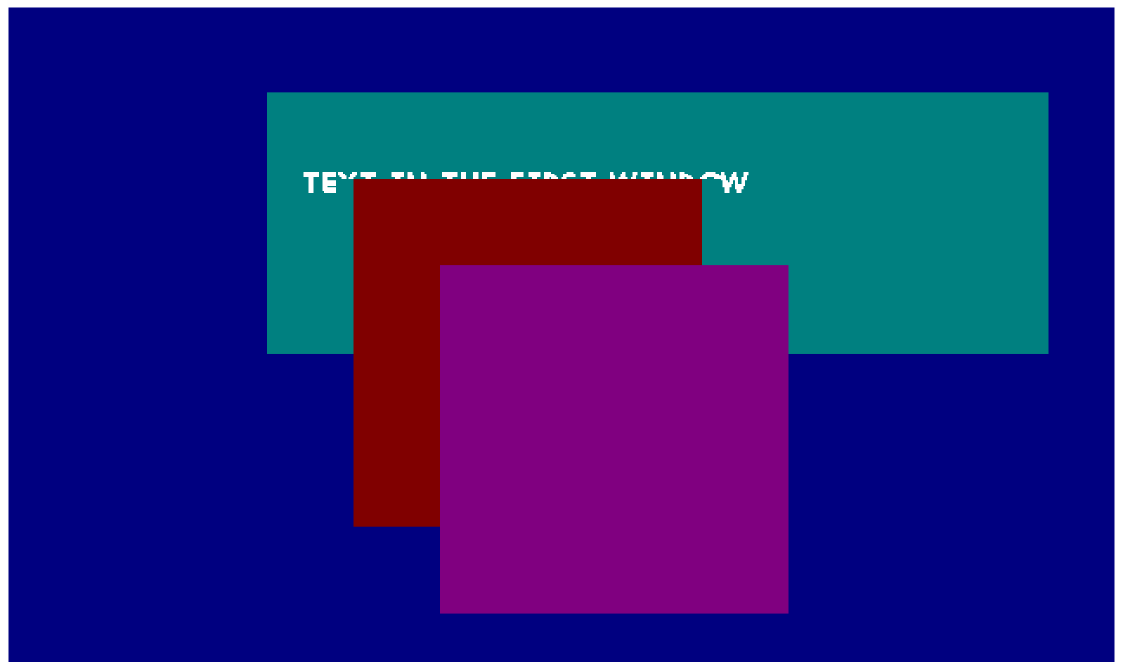
REDRAW SYSTEM TEST PROGRAM



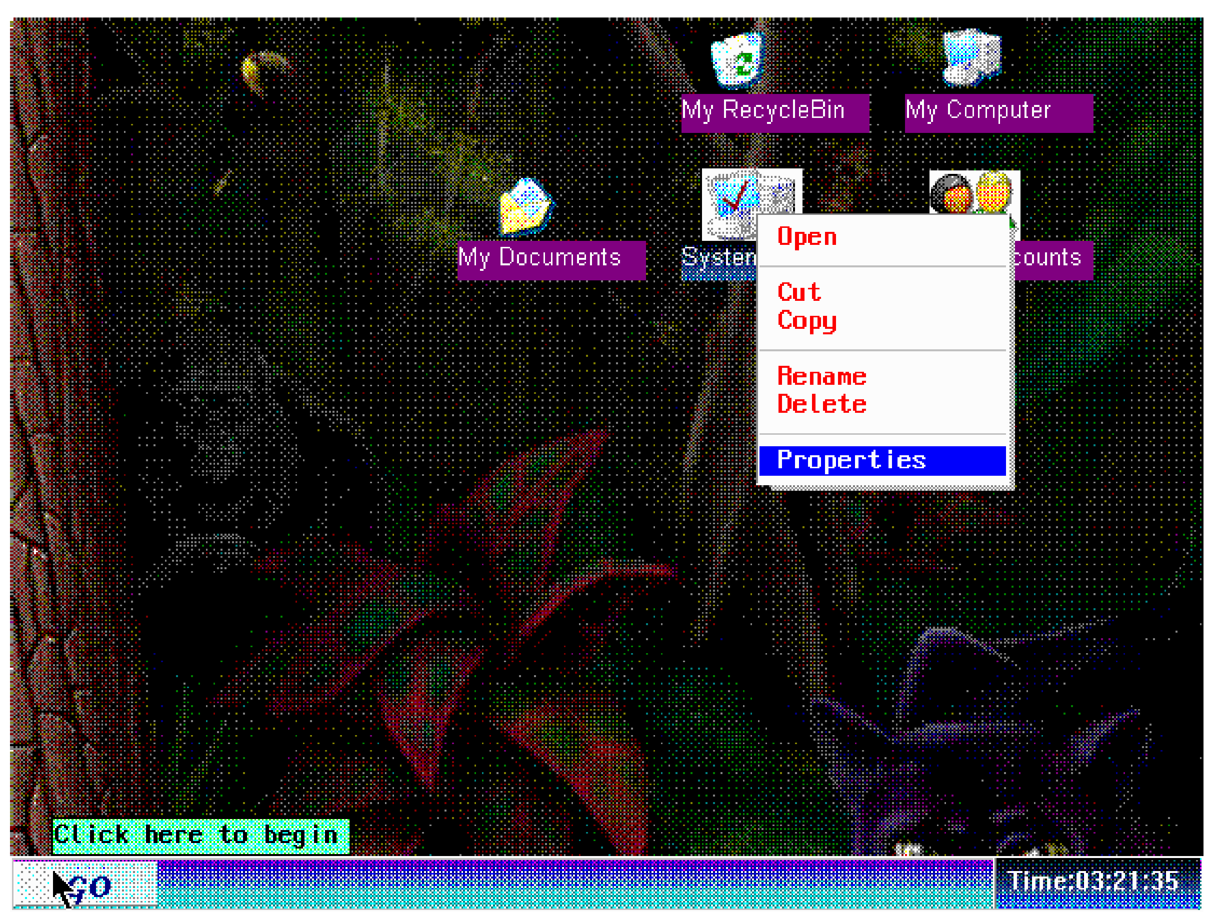
شكل(٢١) – اختبار وحدة إعادة رسم الشاشة اثناء تطوير GUI Package

SUPER GUI DESIGN PROJECT

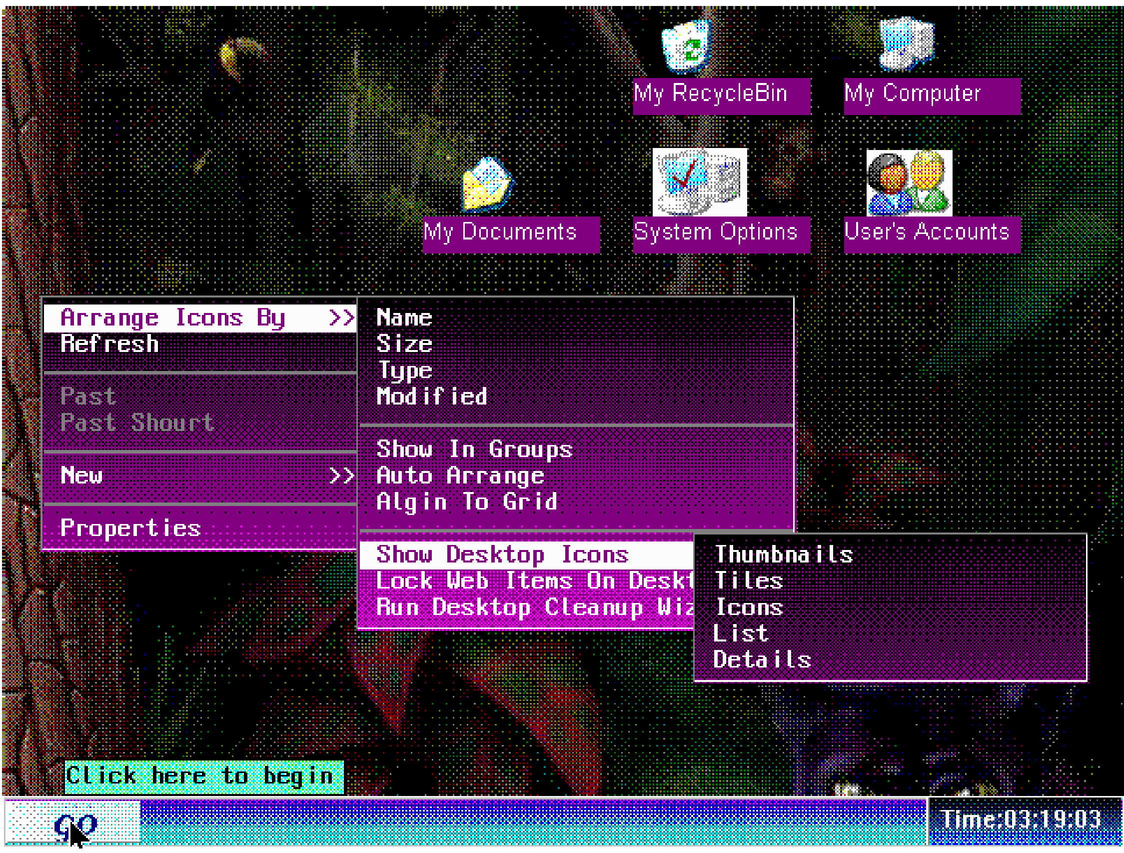
LAYERS SYSTEM TEST PROGRAM



شكل(٢٢) – اختبار وحدة الطبقات اثناء تطوير GUI Package



شكل(٢٣)– اختبار وحدة سطح المكتب اثناء تطوير GUI Package

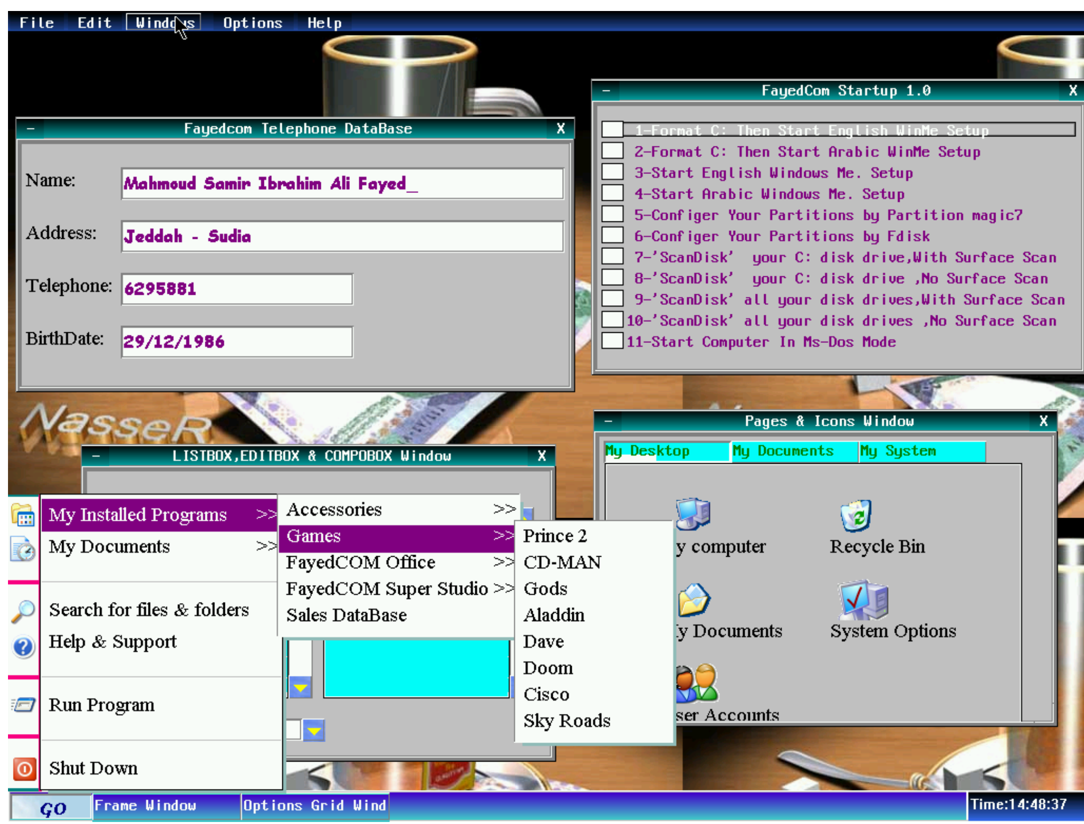


شكل(٢٤)– اختبار وحدة القوائم ذات المستويات المتعددة اثناء تطوير GUI Package

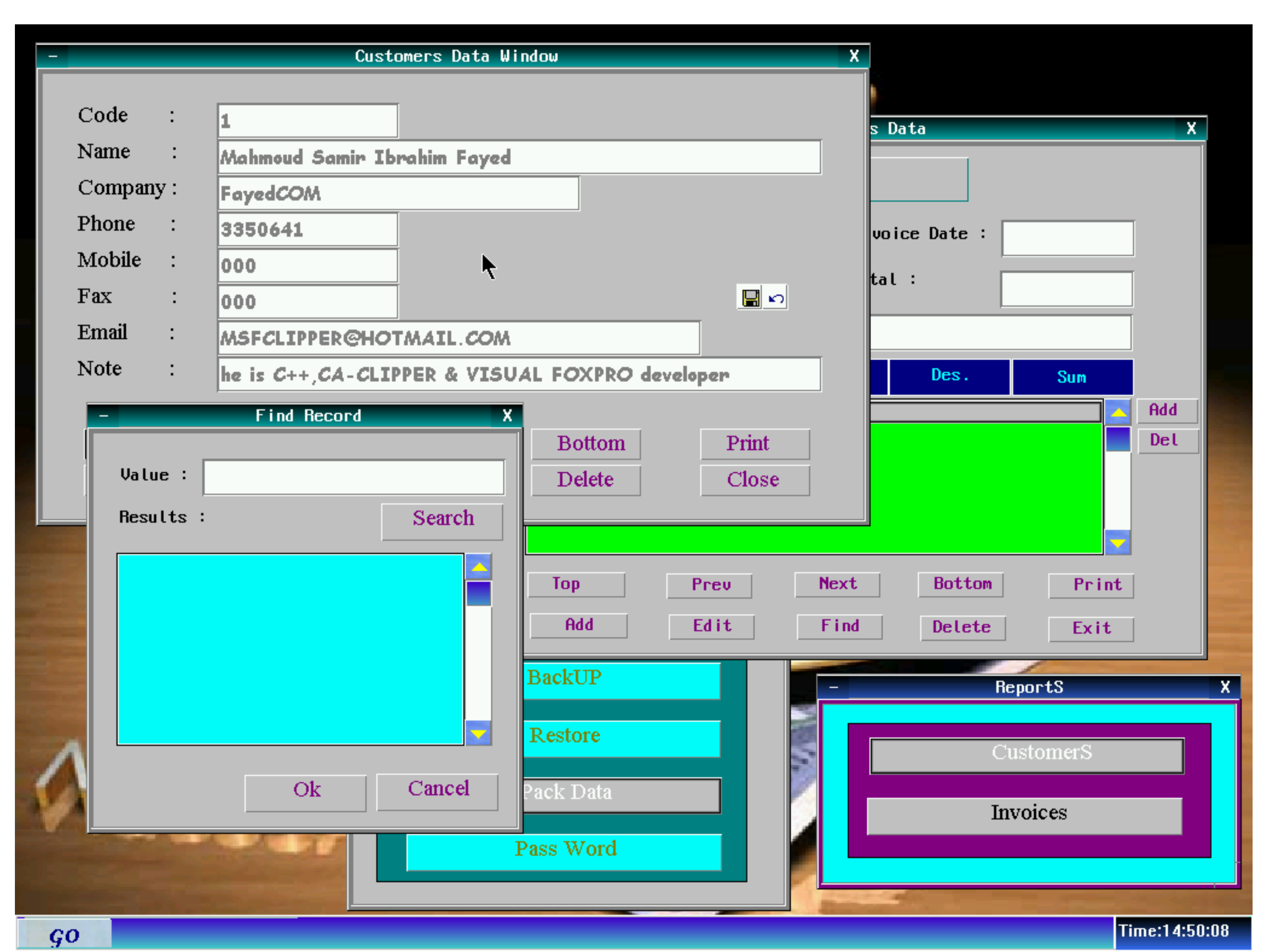
REMOTE CONTROL SYSTEM
VETO SYSTEM
INTERRUPTS SYSTEM
STOP APPLICATION

VETO INT.	:	2
VETO FROM.	:	4
VETO TO	:	3
VETO DATE	:	07/02/2005
VETO TIME	:	19:00:07
VETO R1	:	
VETO R2	:	
VETO R3	:	
VETO R4	:	
VETO R5	:	

شكل(٢٥)– اختبار وحدة التراسل اثناء تطوير GUI Package



شكل(٣٦)– اختبار كل وحدات النظام معا اثناء تطوير GUI Package



شكل(٢٧)– مثال لبرنامج مبيعات Sales باستخدام GUI Package – وهو يعمل تحت DOS

المستوى الرابع لبرمجة واجهة النظام Designer :-

فى هذا المستوى يكون لدينا مصمم يستخدم فى تصميم النماذج بدلا من كتابة التعليمات التى يصحبها الكثير من المجهود حتى نحصل على شكل مناسب – ويختلف كل مصمم نماذج عن الاخر فى الامكانيات التى يوفرها والطريقة التى يعمل بها فبعض برامج التصميم تتيح امكانية تحويل النموذج الذى تم تصميمه الى تعليمات اللغة الاصلية حتى يتم تعديلها بعد ذلك من خلال اى محرر Editor والبعض الاخر لايسمح بذلك.

هناك مصمم نماذج يسمح لك بكتابة التعليمات التى ترتبط بالاحداث – وهناك بعض المصممات لاتسمح بذلك وانما تحول النموذج الى تعليمات اللغة الاصلية ثم بعد ذلك يتم كتابة تعليمات الاحداث داخل Editor.

ان وجود المصمم داخل لغة البرمجة علامة اساسية على مايسمى بـ البصرية Visual التى تتسم بها لغات البرمجة المتطورة.

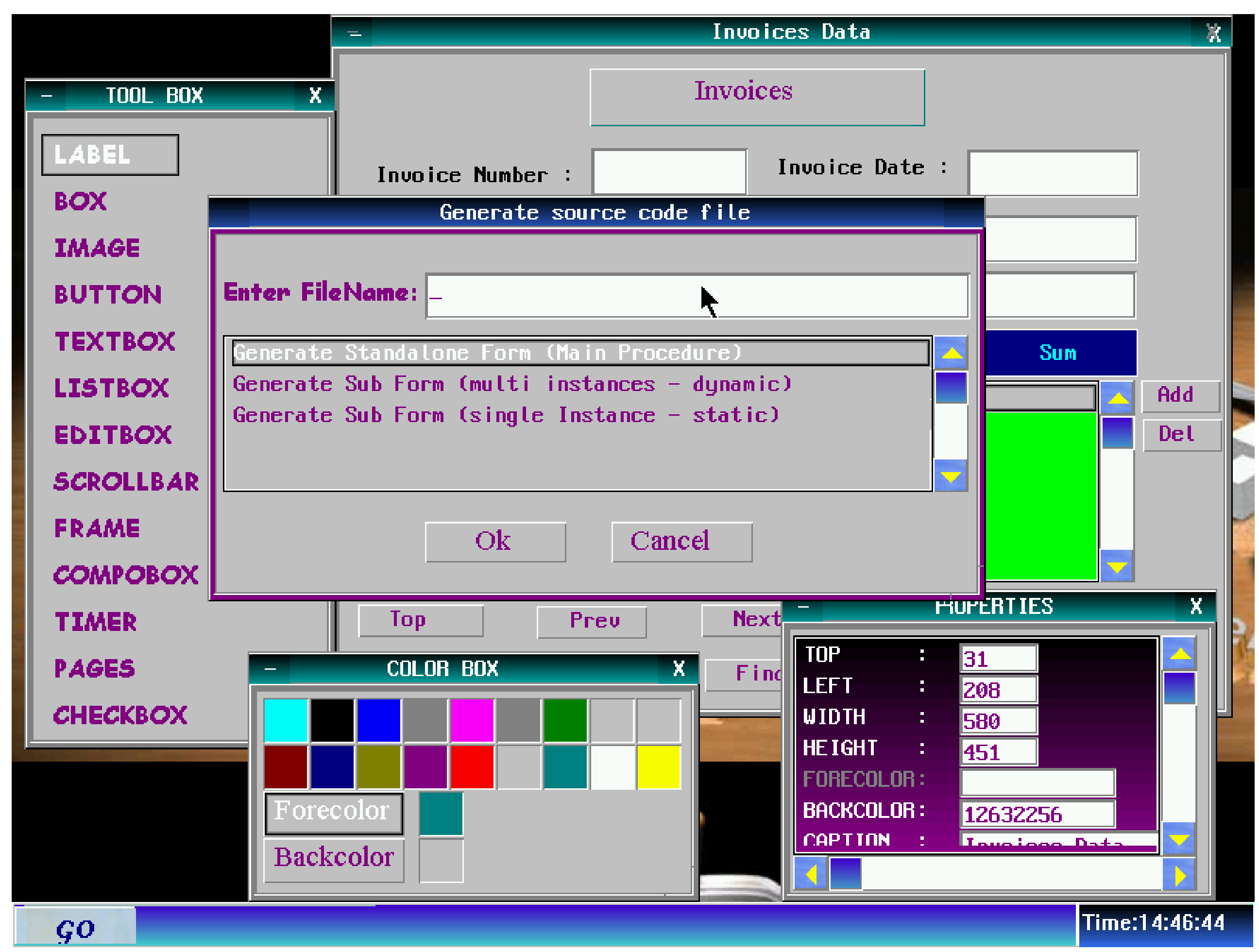
قد تشتمل مكتبة GUI على مصمم Designer وقد لاتشتمل عليه – حيث انه مستوى اخر (مشروع مستقل) فى عالم ادوات تطوير واجهة النظام.

انظر شكل(٢٨) والذى يشتمل على صورة لمصمم نماذج ياتى مع FGLGUI3 والذى قام بتطويره المولف – ان هذا المصمم صغير الحجم فهو لايتعدى 3000 سطر (برمجة هيكلية مع انه يستمل الفصائل فى اداء مهامة اى انه Object Assistence)

شكل(٢٩) يوضح مصمم النماذج الذى تشتمل عليه xHarbour/MiniGUI وهذا المصمم يستخدم مع DoubleS Framework فى تصميم نماذج الخادم Server Forms.

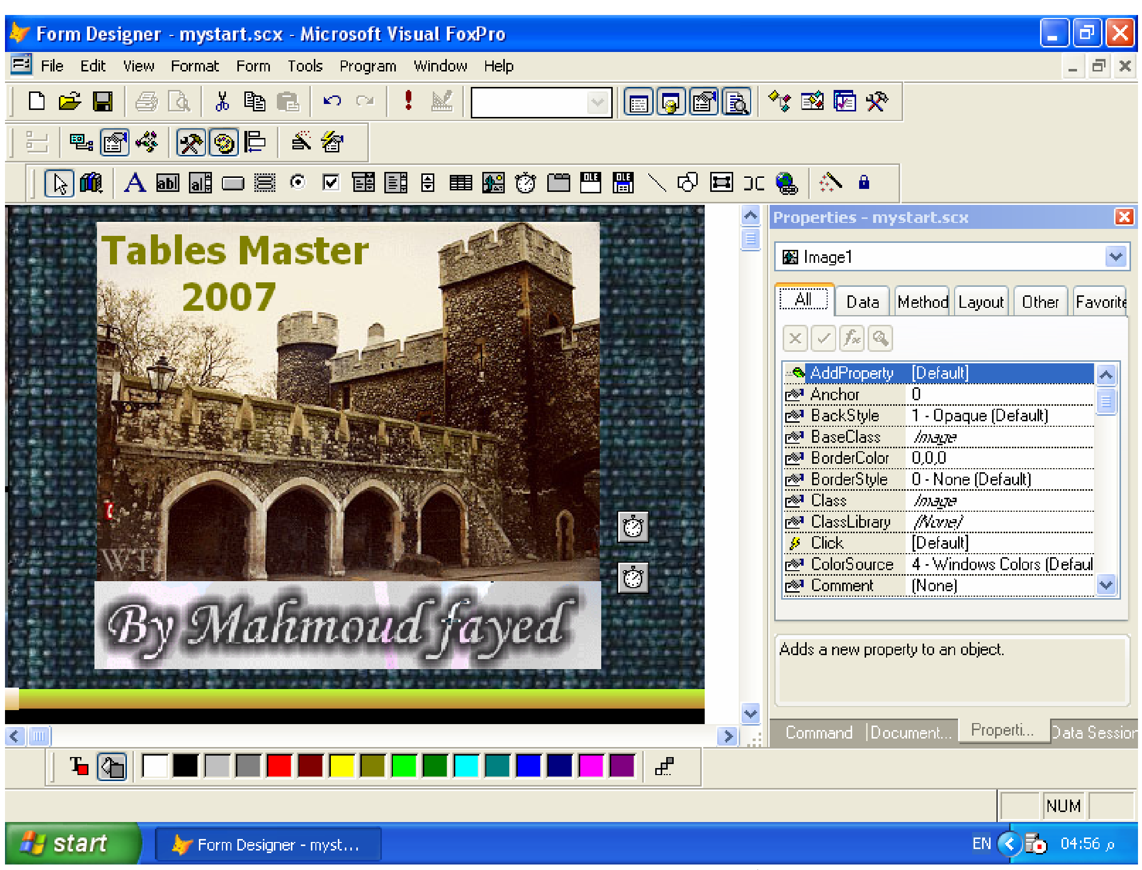
شكل(٣٠) يوضح مصمم نماذج فيجوال فوكس برو ٩ – احدى اشهر لغات البرمجة المستخدمة فى تطوير انظمة قواعد البيانات – وهو احى منتجات شركة Microsoft

شكل(٣١) يوضح مصمم نماذج Visual Studio 2005

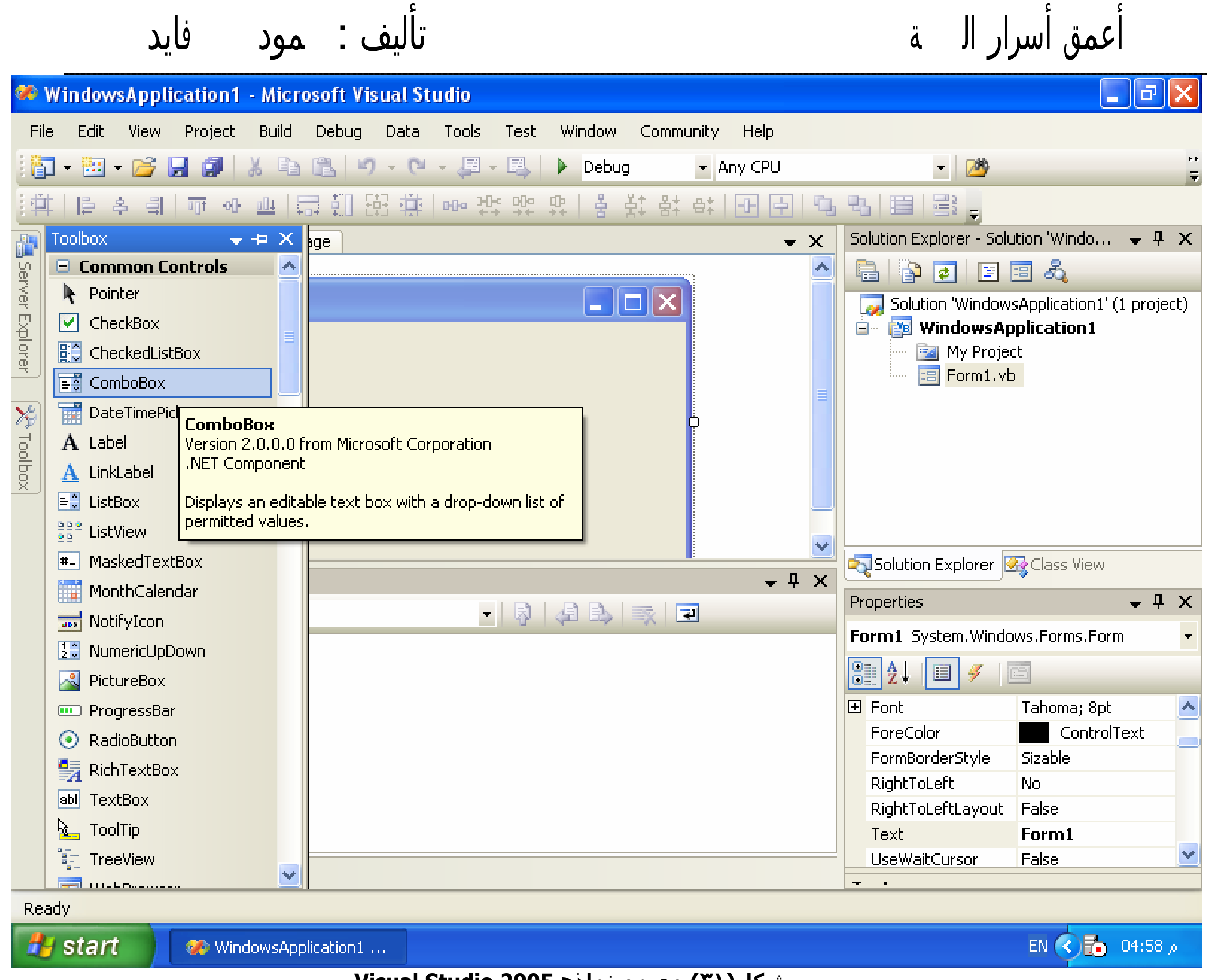


شكل(٢٨) مصمم نماذج تحت نظام DOS ياتى مع المكتبة FGLGUI3 الخاصة بـFGLib الخاصة بلغة CA-Clipper

شكل(٢٩) مصمم xHarbour/MiniGUI المستخدم من قبل DoubleS Framework



شكل(٣٠) مصمم نماذج Microsoft Visual FoxPro 9



شكل(٢١) مصمم نماذج Visual Studio 2005

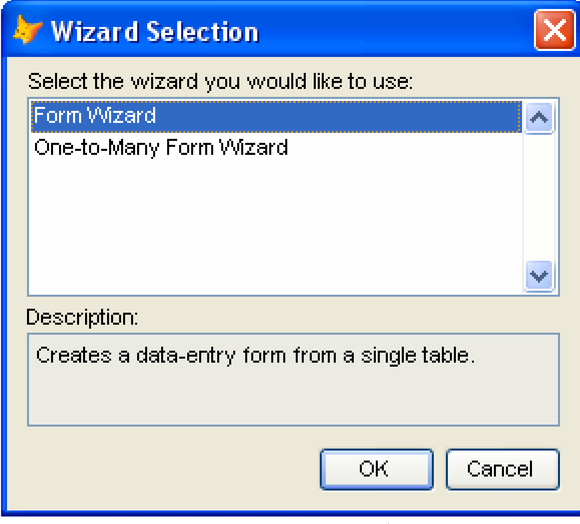
المستوى الخامس لبرمجة واجهة النظام Wizard :-

فى هذا المستوى يتم انتاج الواجهة من خلال المعلومات الازمة الى المعالج Wizard الذى يتولى مهمة انشاء النماذج الازمة ومن هنا يمكن بعد ذلك فى اى وقت تعديل النماذج التى انشاها المعالج لتلائم حاجتنا.

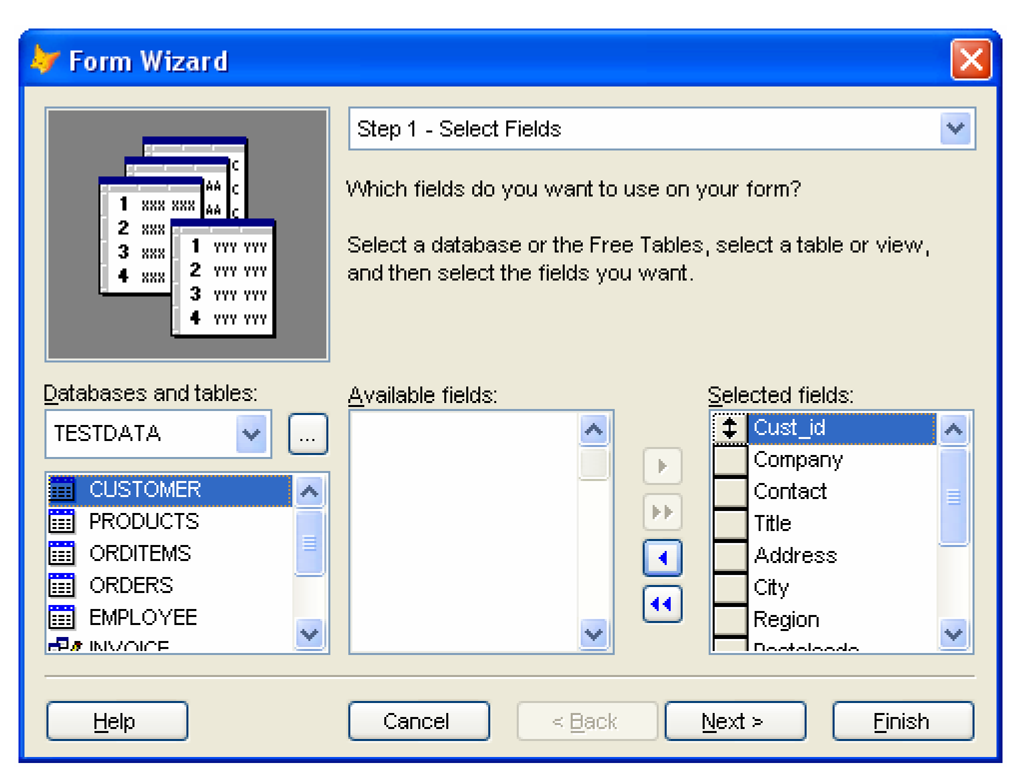
بعض المعالجات لاتنشى ملفات نماذج وانما تنشى ملفات تعليمات او اكواد يمكن تعديلها من خلال المحرر Editor المثال التالى يوضح كيفية استخدام المعالج لتصميم النموذج من خلال فيجوال فوكس برو ٩.

Microsoft Visual FoxPro
File Edit View Format Tools Program Window Help
Wizards
Macros...
Class Browser
Component Gallery
Object Browser
IntelliSense Manager
Task Pane
Toolbox
Beautify...
Task List
Document View
Code References
Coverage Profiler
Breakpoints... Ctrl+B
Debugger
Options...
Table
Query
Form
Report
Label
Mail Merge
PivotTable
Import
Documenting
Upsizing
Application
Database
Web Publishing
Web Services
All Wizards...
Command
MODIFY FORM "d:\documents and
MODIFY FORM "d:\documents and
Command Document... Properti... Data Session
Runs the Form and One-to-Many Form Wizards
NUM
start grbmp Microsoft Visual FoxPro EN 05:23 م

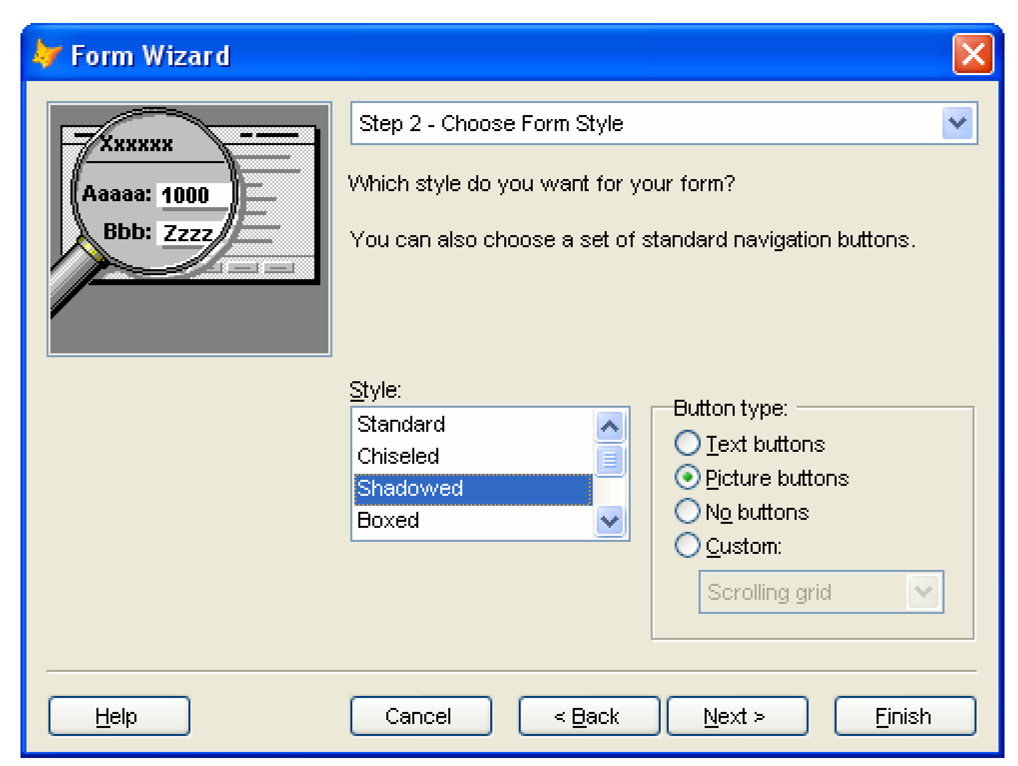
شكل(٣٢) اختيار المعالج المطلوب



شكل(٣٣) اختيار نوع نموذج البيانات



شكل(٣٤) اختيار الحقول التى يشتمل عليها النموذج



شكل(٣٥) اختيار النمط الخاص بالنموذج

Form Wizard

Step 3 - Sort Records

How do you want to sort your records?

Select up to three fields or select one index tag to sort the records by.

Available fields or index tag:

Company
Contact
Title
Address
City
Region
Postalcode

Add >
< Remove
Ascending
Descending

Selected fields:

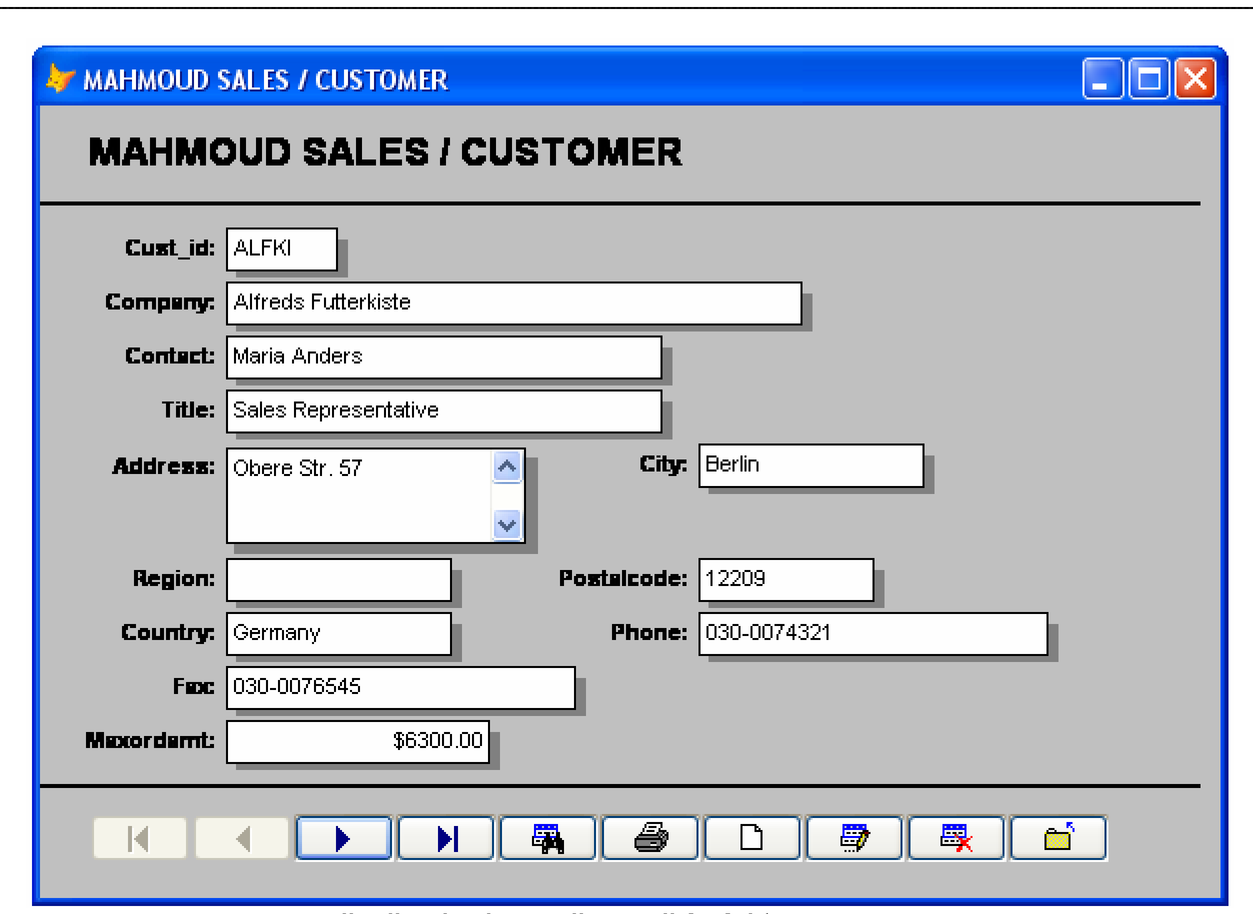
Cust_id

Help | Cancel | < Back | Next > | Finish

شكل(٣٦) اختيار الحقول التى يتم عليها الفهرسة والترتيب



شكل(٣٧) اختيار عنوان النموذج

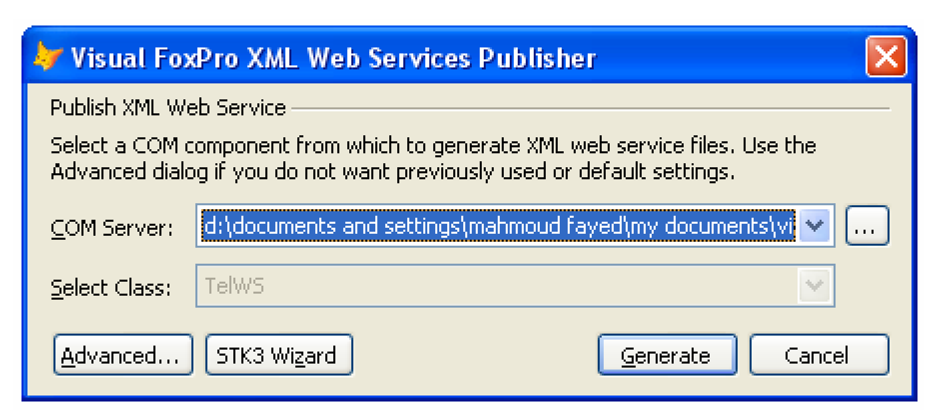


شكل(٣٨) النموذج الذى تم انشائه بالمعالج



المعالج Wizard ليس حكرا على الواجهة والنماذج Forms بل يمتد ليشمل جميع الجوانب مثل الجداول والتقارير وخدمات الويب وهكذا – المثال التالى يوضح ذلك

شكل(٣٩) استخدام المعالج لنشر خدمة ويب Web Services



شكل(٤٠) تحديد COM Component

XML Web Services Publishing Results

Results of components generated:

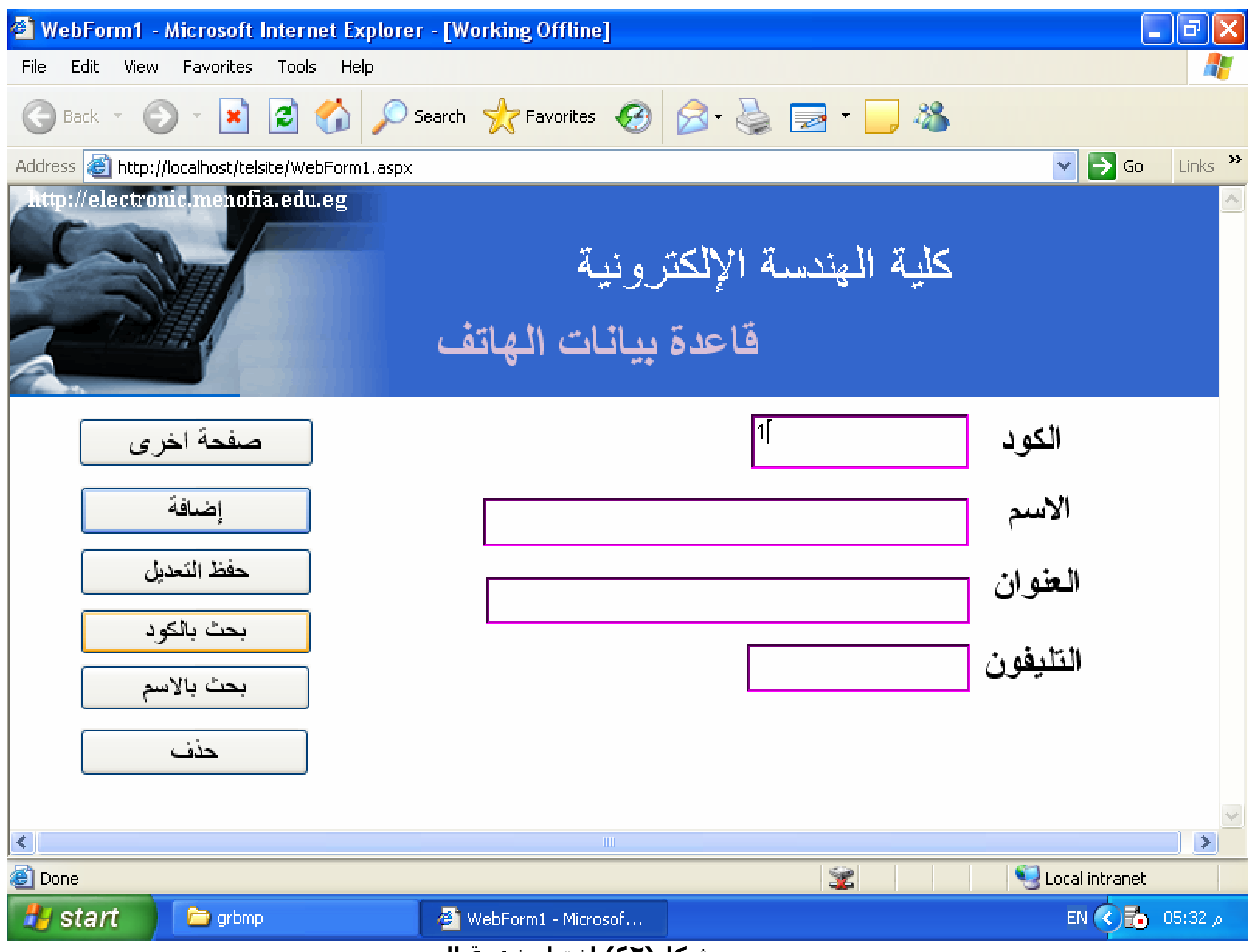
COM Server: d:\documents and settings\mahmoud fayed\my documents\visual foxpro projects\telsite\telsite.dll
Class: TelWS

Generated WSDL: Yes
WSDL: d:\myserver\telsite\TelWS.wsdl
Use ISAPI Listener: Yes
ASP (if not ISAPI): http://FAYEDCOM/TelSite/TelWS.asp

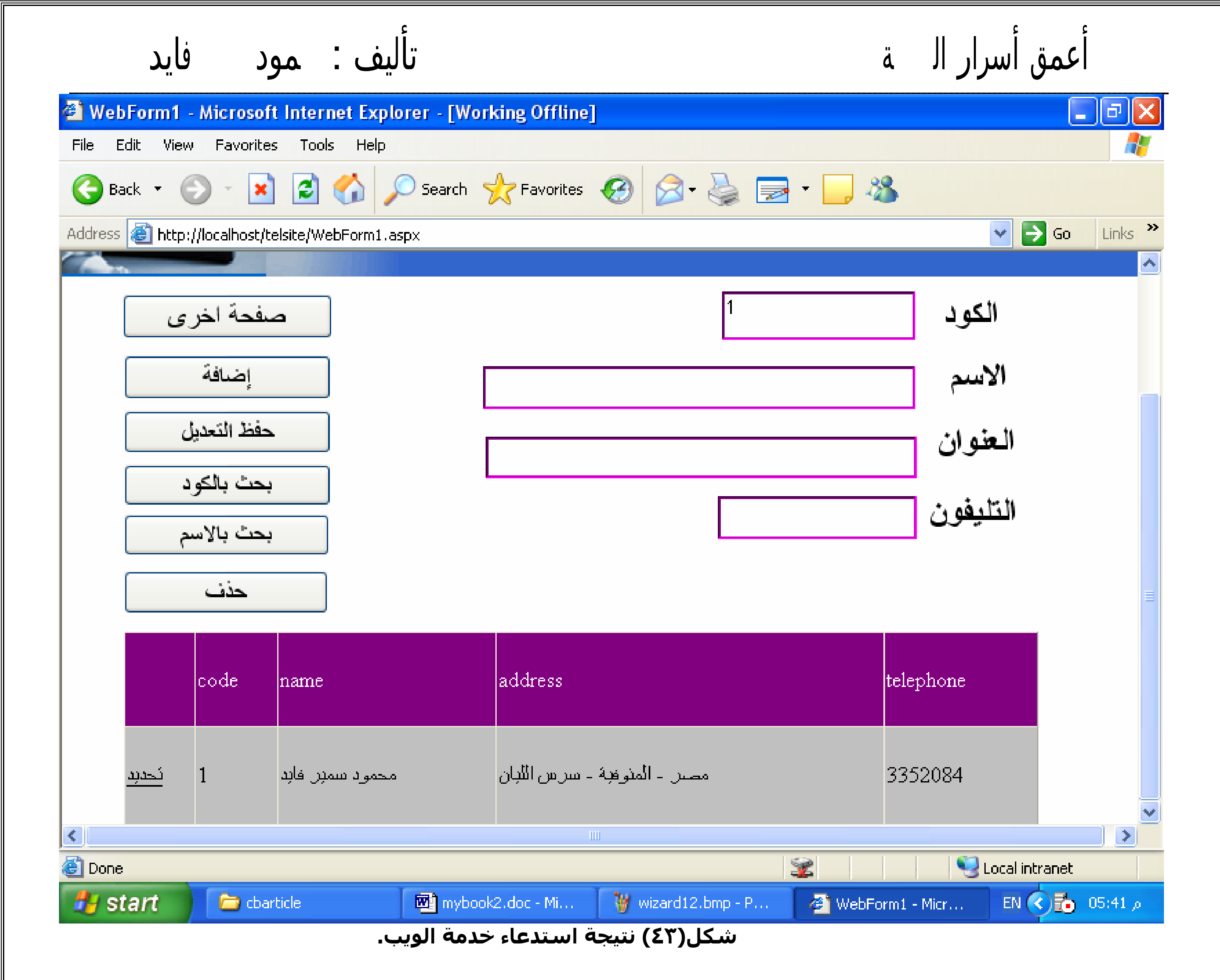
Registered: Yes

OK

شكل(٤١) نتائج عملية النشر



شكل(٤٢) اختبار خدمة الويب



شكل(٤٣) نتيجة استدعاء خدمة الويب.

نهاية الجزء الاول
End of Part(1)

